

ENKBE-2.1

VAX 11/730 8085
TEST PART 4

EP-ENKBE-DL-2.1
FICHE 1 OF 1

MAY 1983
COPYRIGHT © 82-83
MADE IN USA



IDENTIFICATION

Product code: ENKBE VERSION 2.1
Product name: 8085 BASED MICRO-DIAGNOSTIC FOR VAX-11/730 WCS
AND CPU MODULES
Product date: 11-OCT-82
Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC
DECUS
UNIBUS

DECsystem-10
MASSBUS
VAX

DECSYSTEM-20
PDP
VMS

digital

Table of Contents

1.0	ABSTRACT	3
2.0	HARDWARE REQUIREMENTS	3
3.0	SOFTWARE REQUIREMENTS	3
4.0	PREREQUISITES	3
5.0	OPERATING INSTRUCTIONS	3
5.1	Options	3
5.2	Event Flags	4
5.3	APT Setup	4
6.0	PROGRAM FUNCTIONAL DESCRIPTION	4
6.1	Program Overview	4
6.2	Program Size	4
6.3	Program Run Times	4
6.4	Run-time Dynamics	4
6.5	Fault Detection	5
6.6	Performance During Hardware Failures	5
6.7	Program Applications	5
6.8	Test Descriptions	6
7.0	MAINTENANCE HISTORY	6

This program is an 8085 based micro-diagnostic designed to test the hardware on the WCS (Writable Control Store) and DAP (Data Path) modules of the VAX-11/730 processor.

This program will run with a minimum hardware configuration of the WCS and CPU modules present.

Other hardware modules and options may be installed without affecting the diagnostic's performance.

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into the top 3K of 8085 RAM. The micro-diagnostic monitor also resides in 8085 ram, where console software is normally present.

This module should be run after running modules ENKBA thru ENKBD.

This program may be executed by typing DI SE ENKBE after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

Not applicable

Not applicable

Not applicable

The micro-monitor knows when APT is present by the position of the keyswitch and the state of a line connected to the APT-RD port.

Under APT, the diagnostic will halt on error and report error information to APT via the mailbox in 8085 RAM.

The following describes the APT parameters needed to execute this diagnostic:

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. A bottom up diagnostic strategy is utilized which builds on previous tests. The tests should be run in consecutive order, to gain the best isolation of a failing component.

6.2 Program Size

This program runs in 8085 RAM memory and is approximately 3K bytes long.

6.3 Program Run Times

This program will take approximately 25 seconds to run.

6.4 Run-time Dynamics

Not applicable

[illegible]

6.5 Fault Detection

The diagnostic will report errors with the following header:

SECT TST ERR EXP REC OTHER MSK MODULE

1. SECT is the program name (ENKBE)
2. TST is the test number that failed.
3. ERR is the error number that failed.
4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

6.6 Performance During Hardware Failures

A power fail will cause a rebooting of the system (if a battery backup is not present). Other hardware failures will cause normal error printouts.

6.7 Program Applications

This program can be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing component by manufacturing, field service or engineering. The program is APT compatible.

Customers may use the program to verify the basic integrity of the WCS and CPU modules in the system.

6.8 Test Descriptions

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

7.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
8-MAR-82	1.0	Initial release.
28-APR-82	1.01	Turn off run light at end of Test 2D.
11-OCT-82	2.0	RD changes and fault insertion enhancements.
24-FEB-83	2.1	Clear USART CSRs when changing USART modes.

ZZ-ENKBE-2.1

7000

4

H 1

Fiche 1 Frame H1

Sequence 7

ENKBE 8085 based diagnostic for WCS/DAP module Rev 02.00
Table of Contents

7000	3	:	165	DATA AREA
7000	4	:	379	PROGRAM ENTRY POINT
7000	5	:	397	VARIABLES AREA
7000	6	:	496	PROGRAM SPECIFIC SUBROUTINES
7000	7	:	499	CHECK_RD
7000	8	:	525	MASTER_RESET
7000	9	:	538	CLR.CG_PNT
7000	10	:	551	INC.CG_PNT
7000	11	:	564	WRITE_MASTER
7000	12	:	580	WRITE_MODE
7000	13	:	600	WRITE_LOAD
7000	14	:	620	READ_HOLD
7000	15	:	640	WRITE_ALARM
7000	16	:	644	READ_ITDB
7000	17	:	683	WRITE_ITDB
7000	18	:	703	ARM_CNTR
7000	19	:	729	DISARM_CNTR
7000	20	:	756	LOAD_CNTR
7000	21	:	784	SAVE_CNTR
7000	22	:	812	SET_OUTPUT
7000	23	:	829	CLR_OUTPUT
7000	24	:	846	INC_CNTR
7000	25	:	862	WAIT_LOOP
7000	26	:	885	SET_TT_LB
7000	27	:	913	SET_TU_LB
7000	28	:	930	SET_TR_LB
7000	29	:	947	RESTORE_TT
7000	30	:	960	RESTORE_TU
7000	31	:	973	RESTORE_TR
7000	32	:	986	CHECK_REMOTE
7000	33	:	1017	WCS_SIZER
7000	34	:	1093	WCS_SIZE_P
7000	35	:	1114	READ_DATA_16
7000	36	:	1163	EXEC_INST
7000	37	:	1179	LD_CSR_INST
7000	38	:	1218	OR_JMP_ADD
7000	39	:	1270	REPORT_ERROR_2
7000	40	:	1281	REPORT_ERROR
7000	41	:	1292	TEST27_ERROR
7000	42	:	1312	TEST28_ERROR
7000	43	:	1339	TEST29_ERROR
7000	44	:	1360	TEST2A_ERROR
7000	45	:	1379	TEST2A_2_ERROR
7000	46	:	1397	CHECK_2B_2C_ERROR
7000	47	:	1416	TEST2B_2C_ERROR
7000	48	:	1442	TEST2D_ERROR
7000	49	:	1458	INTERVAL_TIMER TESTS
7000	50	:	1459	TEST 27 - COUNTER RUN TEST
7000	51	:	1611	TEST 28 - COUNTER OUTPUT TEST
7000	52	:	1706	TEST 29 - COUNTER RELOAD TEST
7000	53	:	1953	TEST 2A - COUNTER CONCATENATION TEST
7000	54	:	2167	TEST 2B - SUMMARY REGISTER TEST (TIMER INPUTS)
7000	55	:	2473	TEST 2C - SUMMARY REGISTER TEST (USART INPUTS)
7000	56	:	2945	TEST 2D - INCREMENT THROUGH WCS TEST
7000	57	:		
7000	58	:		
7000	59	:		
7000	60	:		
7000	61	:		
7000	62	:		

7000 63
7000 64
7000 65
7000 66
7000 67
7000 68
7000 69
7000 70
7000 71
7000 72
7000 73
7000 74
7000 75
7000 76
7000 77
7000 78
7000 79
7000 80
7000 81
7000 82
7000 83
7000 84
7000 85
7000 86
7000 87
7000 88
7000 89
7000 90
7000 91
7000 92
7000 93
7000 94
7000 95
7000 96
7000 97
7000 98
7000 99
7000 100
7000 101
7000 102
7000 103
7000 104
7000 105
7000 106
7000 107
7000 108
7000 109
7000 110
7000 111
7000 112
7000 113
7000 114
7000 115
7000 116
7000 117
7000 118
7000 119
7000 120
7000 121
7000 122

TITLE "ENKBE 8085 based diagnostic for WCS/DAP module Rev 02.10"

COPYRIGHT (c) 1982 BY
DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS. ALL RIGHTS RESERVED.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER COPIES THEREOF
MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON. NO
TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

++

FACILITY: VAX-11/730 MICRO-DIAGNOSTIC.

ABSTRACT: This 8085 based diagnostic tests the hardcore logic of the
WCS and DAP modules.

ENVIRONMENT: ENKAA Micro-diagnostic Monitor

AUTHOR: Jim Klumpp 4-MAY-81

MODIFIED BY: David Mayo 8-MAR-82 VERSION 01.00
Initial release.

David Mayo 16-APR-82 VERSION 01.10
Turn off run light at end of Test 2D.

David Mayo 8-AUG-82 VERSION 01.20
Change Test 2C to skip terminal USART testing
if under RD. Reset terminal USART to 35H after
testing.

Mike Ricchetti 10-SEP-82 VERSION 02.00
Added second part to Test 29 to check that counter
group 1 can be reloaded when the timer is used in
the gate control mode.

Mike Ricchetti 27-SEP-82
Add check for new interval timer signal 'WCSB TIMER INTR H'
in Test 2B to see that it interrupts 8085.

Ernie Preisig 30-SEP-82
Added code to skip interval timer tests when under RD due
to timing constraints. Added code to allow test 2C AND 2D
to run under RD (only the TU58 UART is tested under RD in
test 2C). Shortened test code (via subroutines) to allow

ZZ-ENKBE-2.1 7000 4

7000 123
7000 124
7000 125
7000 126
7000 127
7000 128
7000 129
7000 130
7000 131
7000 132
7000 138
7000 139
7000 140
7000 141
7000 142
7000 143
7000 144
7000 145
7000 146
7000 147
7000 148
7000 149
7000 150
7000 151
7000 152
7000 153
7000 154
7000 155
7000 156
7000 157
7000 158
7000 159
7000 160
7000 161
7000 162
7000 163
7000 164
7000 165
7000 166
7000 167
7000 168 000
7000 169
7000 170
7000 171
7000 172
7000 173
7000 174
7000 175
7000 176
7000 177
7000 178 00000000
7000 179
7000 180 00000000
00000002
00000004
00000006
00000006
00000001
00000003
00000005

.....
:--

J 1

Fiche 1 Frame J1

Sequence 9

room for modifications.

Mike Ricchetti 5-OCT-82

Check for 100 HZ signal in Test 2A.

PETER GREEN 24-FEB-83 VERSION 2.10

Added code to clear all USART CSRs when setting USART to
loopback mode and when restoring USART to normal mode.

.PSECT CPU5,BYTE
.ALIGN BYTE

: DATA AREA
: *****

HEAD

=0

EQUATE

;REGISTER EQUATE MACRO

ZZ-ENKBE-2.1 7000 4
00000007 00000006

K 1

Fiche 1 Frame K1

Sequence 10

7000 181				
7000 182	00000000	UPC14A	=<^X00>	;(RO) UPC BIT 14 ASSERTED HIGH <MSB>
7000 183				
7000 184				
7000 185	00000001	MISC2	=<^X01>	
7000 186	00000001	BOOTF	=<^X01>	;(RO) BOOT FLAG. ASSERTED LOW. <LSB>
7000 187				
7000 188				
7000 189	00000002	DCLO	=<^X02>	;(RO) DC LOW FLAG ASSERTED HIGH<MSB>
7000 190				
7000 191	00000002	HLTFLG	=<^X02>	;(RO) HALT FLAG ASSERTED LOW <LSB>
7000 192				
7000 193				
7000 194	00000003	READJ1	=<^X03>	;(RO) JUMPER J1 FLAG ASS. HIGH <MSB>
7000 195				
7000 196	00000003	ALLOWP	=<^X03>	;(RO) ENABLE CONTROL P ASS LOW <LSB>
7000 197				
7000 198				
7000 199	00000004	READJ2	=<^X04>	;(RO) BIT 07 EQUALS J2 <MSB>
7000 200	00000004	APTFLG	=<^X04>	;(RO) APT PRESENT FLAG <LSB>
7000 201				
7000 202	00000005	AUTO	=<^X05>	;(RO) AUTO TEST FLAG ASS LOW <LSB>
7000 203				
7000 204	00000006	LAT100HZ	=<^X06>	;(RO) 100 HZ CLOCK SIGNAL <MSB>
7000 205	00000006	REMDIS	=<^X06>	;(RO) REMOTE DISABLE FLAG ASS LOW <LSB>
7000 206				
7000 207	00000007	OKV15	=<^X07>	;(RO) 15 VOLTS OK FLAG ASS LOW <MSB>
7000 208				
7000 209	00000007	BCOTEN	=<^X07>	;(RO) BOOT ENABLE FLAG ASS LOW <LSB>
7000 210				
7000 211				
7000 212	00000008	WCSB0	=<^X08>	;(WO) WCS WRITE DATA REGISTER
7000 213	00000009	WCSB1	=<^X09>	
7000 214	0000000A	WCSB2	=<^X0A>	
7000 215				
7000 216	0000001C	ITDB	=<^X1C>	;(R/W) TIMER DATA REGISTER
7000 217	0000001D	ITCSR	=<^X1D>	;(R/W) TIMER STATUS REGISTER
7000 218				
7000 219	00000020	SUMREG	=<^X20>	;(RO) SUMMARY REGISTER
7000 220				
7000 221	00000020	CLRCLK	=<^X20>	;(WO) CLEAR CPU CLOCK RUN
7000 222	00000021	SETCLK	=<^X21>	;(WO) SET CPU CLOCK RUN
7000 223				
7000 224	00000022	CLRSS	=<^X22>	;(WO) CLEAR CLOCK SINGLE STEP
7000 225	00000023	SETSS	=<^X23>	;(WO) SET CLOCK SINGLE STEP
7000 226				
7000 227	00000024	CLRCSK	=<^X24>	;(WO) CLEAR CSR CLOCK
7000 228	00000025	SETCSK	=<^X25>	;(WO) SET CSR CLOCK
7000 229				
7000 230	00000026	CLRUPK	=<^X26>	;(WO) CLEAR THE UPC CLOCK
7000 231	00000027	SETUPK	=<^X27>	;(WO) SET THE UPC CLOCK
7000 232				
7000 233	00000028	SETMCI	=<^X28>	;(WO) SET MEMORY CONTROL INIT
7000 234	00000029	CLRMCI	=<^X29>	;(WO) CLEAR MEMORY CONTROL INIT
7000 235				
7000 236	0000002A	SETMCK	=<^X2A>	;(WO) SET MEMORY CONTROL CLOCK
7000 237	0000002B	CLRMCK	=<^X2B>	;(WO) CLEAR MEMORY CONTROL CLOCK
7000 238				
7000 239	0000002C	CLRTMR	=<^X2C>	;(WO) STOP TIMER

ZZ-ENKBE-2.1 7000 4			L 1	Fiche 1 Frame L1	Sequence 11
7000	240	0000002D	SETTMR	=<^X2D>	;(WO) START TIMER COUNTING
7000	241				
7000	242	0000002E	CLRBSY	=<^X2E>	;(WO) CLEAR UNIBUS BUS BUSY
7000	243	0000002F	SETBSY	=<^X2F>	;(WO) SET UNIBUS BUS BUSY
7000	244				
7000	245	00000030	SETDCL	=<^X30>	;(WO) SET DC LOW
7000	246	00000031	CLRDCL	=<^X31>	;(WO) CLEAR DC LOW
7000	247				
7000	248	00000032	SETACL	=<^X32>	;(WO) SET AC LOW
7000	249	00000033	CLRACL	=<^X33>	;(WO) CLEAR AC LOW
7000	250				
7000	251	00000034	INITL	=<^X34>	;(WO) CLEAR UNIBUS INIT
7000	252	00000035	INITH	=<^X35>	;(WO) SET UNIBUS INIT
7000	253				
7000	254	00000036	CLRWCK	=<^X36>	;(WO) CLEAR WCS BIT 00
7000	255	00000037	SETWCK	=<^X37>	;(WO) SET WCS BIT 00
7000	256				
7000	257	00000038	CLRCR	=<^X38>	;(WO) CLEAR WCS BIT 01
7000	258	00000039	SETCSR	=<^X39>	;(WO) SET WCS BIT 01
7000	259				
7000	260	0000003C	CLRLIT	=<^X3C>	;(WO) CLEAR THE RUN LIGHT
7000	261	0000003D	SETLIT	=<^X3D>	;(WO) SET THE RUN LIGHT
7000	262				
7000	263	00000040	TRDB	=<^X40>	;(R/W) REMOTE USART DATA BUFFER
7000	264	00000041	TRSR	=<^X41>	;(RO) REMOTE USART STATUS REGISTER
7000	265	00000042	TRMODE	=<^X42>	;(R/W) REMOTE MODE REGISTER 1 + 2
7000	266				;(FIRST ACCESS GET MR1, SECOND GETS MR2
7000	267				;(READING THE CR FORCES THE INTERNAL
7000	268				;(POINTER BACK TO MR1
7000	269	00000043	TRCR	=<^X43>	;(R/W) REMOTE USART COMMAND REGISTER
7000	270				
7000	271	00000044	TUDB	=<^X44>	;(R/W) TU-58 DATA BUFFER
7000	272	00000045	TUSR	=<^X45>	;(RO) TU-58 STATUS REGISTER
7000	273	00000046	TUMODE	=<^X46>	;(R/W) TU-58 MODE REGISTER 1 + 2
7000	274				;(FIRST ACCESS GET MR1, SECOND GETS MR2
7000	275				;(READING THE CR FORCES THE INTERNAL
7000	276				;(POINTER BACK TO MR1
7000	277	00000047	TUCR	=<^X47>	;(R/W) TU-58 COMMAND REGISTER
7000	278				
7000	279	00000048	TTDB	=<^X48>	;(R/W) TERMINAL DATA BUFFER
7000	280	00000049	TTSR	=<^X49>	;(RO) TERMINAL STATUS REGISTER
7000	281	0000004A	TTMODE	=<^X4A>	;(R/W) TERMINAL MODE REGISTER 1 + 2
7000	282				;(FIRST ACCESS GET MR1, SECOND GETS MR2
7000	283				;(READING THE CR FORCES THE INTERNAL
7000	284				;(POINTER BACK TO MR1
7000	285	0000004B	TTCR	=<^X4B>	;(R/W) TERMINAL COMMAND REGISTER
7000	286				
7000	287	00000080	READ	=<^X80>	;(RO) CONSOLE READ REGISTER
7000	288				
7000	289	00000082	CPUACK	=<^X82>	;(RO) ACKNOWLEDGE FROM CPU <LSB>
7000	290				
7000	291	00000083	CPATTN	=<^X83>	;(RO) ATTENTION FROM CPU <LSB>
7000	292				
7000	293	00000084	UPC14	=<^X84>	;(RO) UPC BIT 14 <LSB>
7000	294				
7000	295	00000085	CSR07	=<^X85>	;(RO) CSR BIT 7 <LSB>
7000	296				
7000	297	00000086	CSR15	=<^X86>	;(RO) CSR BIT 15 <LSB>
7000	298				
7000	299	00000087	CSR23	=<^X87>	;(RO) CSR BIT 23 <LSB>

ZZ-ENKBE-2.1		7000	4	M 1	Fiche 1 Frame M1	Sequence 12
7000	300					
7000	301	000000A0		ENBPAP	=<^XA0>	;(WO) ENABLE STALL ON PARITY ERROR
7000	302	000000A1		DISPAR	=<^XA1>	;(WO) DISABLE STALL ON PARITY ERROR
7000	303					
7000	304	000000A2		VSHIFT	=<^XA2>	;(WO) SET THE V-BUS TO SHIFT MODE
7000	305	000000A3		VNORML	=<^XA3>	;(WO) SET THE V-BUS TO NORMAL MODE
7000	306					
7000	307	000000A4		SETTIM	=<^XA4>	;(WO) SET THE INTERVAL TIMER INTERRUPT
7000	308	000000A5		CLRTIM	=<^XA5>	;(WO) CLEAR THE INTERVAL TIMER INTRPT.
7000	309					
7000	310	000000A6		ENBMEM	=<^XA6>	;(WO) ENABLE MEMORY REFERENCES
7000	311	000000A7		DISMEM	=<^XA7>	;(WO) DISABLE MEMORY REFERENCES
7000	312					
7000	313	000000A8		SETACK	=<^XA8>	;(WO) SET CONSOLE ACKNOWLEDGE
7000	314	000000A9		CLRACK	=<^XA9>	;(WO) CLEAR CONSOLE ACKNOWLEDGE
7000	315					
7000	316			;NOTE: THE CONSOLE ACKNOWLEDGE SIGNAL IS ALSO USED AS THE UPC		
7000	317			;SERIAL INPUT. WHEN USED THIS WAY THE SENSE IS INVERTED. TO SIMPLIFY		
7000	318			;MATTERS THE NEXT TWO EQUATES ARE INCLUDED		
7000	319					
7000	320	000000A8		CLRUPC	=<^XA8>	;(WO) CLEAR THE V-BUS SERIAL INPUT
7000	321	000000A9		SETUPC	=<^XA9>	;(WO) SET THE V-BUS SERIAL INPUT
7000	322					
7000	323	000000AA		SETATN	=<^XAA>	;(WO) SET CONSOLE ATTENTION
7000	324	000000AB		CLRATN	=<^XAB>	;(WO) CLEAR CONSOLE ATTENTION
7000	325					
7000	326	000000AC		SETPFI	=<^XAC>	;(WO) SET POWER FAIL INTERRUPT
7000	327	000000AD		CLRPFI	=<^XAD>	;(WO) CLEAR POWER FAIL INTERRUPT
7000	328					
7000	329	000000AE		SETHLT	=<^XAE>	;(WO) SET THE CONSOLE HALT BIT
7000	330	000000AF		CLRHLT	=<^XAF>	;(WO) CLEAR THE CONSOLE HALT BIT
7000	331					
7000	332	000000CC		WRITE	=<^XCC>	;(WO) THE CONSOLE WRITE REGISTER
7000	333					
7000	334	000000EC		YBUSRD	=<^XEC>	;(RO) Y-BUS READ
7000	335					
7000	336					
7000	337					
7000	338			;ADDITIONAL DATA		
7000	339					
7000	340	000040D9		MIPFLG	= <^X40D9>	;ADDRESS OF MODEM IN PROGRESS FLAG.
7000	341	00004C22		PARITY VECTOR	= <^X4C22>	;PARITY ERROR BRANCH VECTOR.
7000	342	00004C25		POWER VECTOR	= <^X4C25>	;POWER FAIL BRANCH VECTOR.
7000	343	000000FF		WAIT_ATTEN_CNT	= <^XFF>	;COUNT FOR WAIT CPU ATTN LOOP.
7000	344					
7000	345					
7000	346			;INTERVAL TIMER DATA		
7000	347					
7000	348	000000FF		RESET DATA	= <^XFF>	;EXECUTE A MASTER RESET.
7000	349	00000017		MASTER MODE	= <^X17>	;POINT TO THE MASTER MODE REG.
7000	350	00000000		MODE_REG	= <^X00>	;POINT TO A COUNTER MODE REG.
7000	351	00000008		LOAD_REG	= <^X08>	;POINT TO A COUNTER LOAD REG.
7000	352	00000010		HOLD_REG	= <^X10>	;POINT TO A COUNTER HOLD REG.
7000	353	00000007		ALARM_REG1	= <^X07>	;POINT TO ALARM REG 1.
7000	354	0000000F		ALARM_REG2	= <^X0F>	;POINT TO ALARM REG 2.
7000	355	00000020		ARM_CNTR_DAT	= <^X20>	;ARM A COUNTER.
7000	356	000000C0		DISARM_CNTR_DAT	= <^XC0>	;DISARM A COUNTER.
7000	357	00000040		LOAD_CNTR_DAT	= <^X40>	;LOAD A COUNTER.
7000	358	000000A0		SAVE_CNTR_DAT	= <^XA0>	;SAVE A COUNTER'S CONTENTS.
7000	359	000000E8		SET_OUTPUT_DAT	= <^XE8>	;ASSERT A COUNTER'S OUTPUT.


```

ZZ-ENKBE-2.1 7000 4
7000 360 000000E0
7000 361 000000F0
7000 362 0000003F
7000 363 00000045
7000 364 00000050
7000 365 000000C0
7000 366 000000C0
7000 367 00000040
7000 368
7000 369
7000 370
7000 371
7000 372 000000B3
7000 373 00000015
7000 374 00000035
7000 375 00000002
7000 376 00000001
7000 377 00000008
7000 378 00000001
7000 379 00000010
7000 380 00000004
7000 381 00000020
7000 382 00000002
7000 383
7000 384
7000 385
7000 386
7000 387
7000 388
7000 389
7000 390
7000 391 045E'C3
7003 392
7003 393
7003 394
7003 395
7003 396
7003 397
7003 398
7003 399
7003 400 0210
7005 401 45 42
7007 402
7007 403
7007 404
7007 405
7007 406
7007 407
7007 408
7007 409 0000
7009 410 00
700A 411 00
700B 412 00
700C 413 00
700D 414 00
700E 415 0000
7010 416 00
7011 417 00
7012 418 0000
7014 419 0000

```

N 1

Fiche 1 Frame N1 Sequence 13

```

CLR_OUTPUT_DAT = <^XE0>
INC_CNTR_DAT   = <^XF0>
ARM_ALL        = <^X3F>
WAIT_CNT1      = <^X45>
WAIT_CNT2      = <^X50>
WAIT_CNT3      = <^XC0>
SET_SER_OUT    = <^XC0>
CLR_SER_OUT    = <^X40>

```

```

;CLEAR A COUNTER'S OUTPUT.
;SINGLE STEP A COUNTER.
;ARM ALL COUNTERS.
;COUNT FOR WAIT LOOP.
;*
;*
;SET THE 8085 SERIAL OUTPUT.
;CLEAR THE 8085 SERIAL OUTPUT.

```

;USART DATA

;DATA TO...

```

UA_LOOPBACK    = <^XB3>
UA_RESTORE     = <^X15>
UATT_RESTORE   = <^X35>
REC_DONE       = <^X02>
TX_RDY         = <^X01>
TER_REC_BIT    = <^X08>
TER_TX_BIT     = <^X01>
TUS8_REC_BIT   = <^X10>
TUS8_TX_BIT    = <^X04>
REM_REC_BIT    = <^X20>
REM_TX_BIT     = <^X02>

```

```

;PUT A USART IN LOCAL LOOPBACK MODE.
;RESTORE USART TO NORMAL MODE.
;RESTORE TERMINAL USART TO NORMAL MODE.
;READ RECEIVE DONE SIGNAL.
;READ TRANSMIT READY.
;READ TEMINAL RECEIVE READY.
;READ TERMINAL TRANSMIT READY.
;READ TUS8 RECEIVE READY.
;READ TUS8 TRANSMIT READY.
;READ REMOTE RECEIVE READY.
;READ REMOTE TRANSMIT READY.

```

```

;*****
; PROGRAM ENTRY POINT.... MUST BE THE FIRST INSTUCTION IN THE PROGRAM
;*****

```

ENTRY: JMP TEST27

```

;*****
; VERSION NUMBER AND LAST TWO LETTERS OF FILE NAME. THESE 2 WORDS CAN NOT
; BE MOVED TO ANY OTHER LOCATION WITHOUT A MICMON CHANGE.
;*****

```

```

VERSION: .WORD <^X0210> ;VERSION NUMBER (REV 02.00)
FILE_NAME: .ASCII 'BE' ;LAST 2 CHARS OF FILE NAME

```

```

;*****
; VARIABLES AREA
;*****

```

```

SAVE_WCS_ADD: .WORD 0 ;SAVES WCS ADDRESS ON ^C USE.
NO_CPU_ATTN: .BYTE 0 ;FLAG=1, NO CPU ATTN RECEIVED.
SKIP-UA1: .BYTE 0 ;FLAG=1, DON'T TEST USART1.
SKIP-UA3: .BYTE 0 ;FLAG=1, DON'T TEST USART3.
CG_PNT: .BYTE 0 ;POINTS TO COUNTER GROUP.
SHIFT_CNT: .BYTE 0 ;COUNTS ACCUMULATOR SHIFTS.
TIMER_DAT: .WORD 0 ;DATA TO BE SENT TO TIMER.
WAIT_CNT: .BYTE 0 ;WAIT LOOP COUNT.
COUNT_SAVE: .BYTE 0 ;TEMPORARY COUNTER STORAGE.
TEMP_WORD: .WORD 0 ;TEMP STORAGE LOCATION.
WCS_ADDR_PNT: .WORD 0 ;POINTS TO WCS_SIZER_ADDR.

```

```

ZZ-ENKBE-2.1 7000 4
7016 420 0000
7018 421
7018 422 003C
701A 423 0040
701C 424 0044
701E 425 0048
7020 426 004C
7022 427
7022 428 20
7023 429 48 47 49 48 00'
      57 20 54 53 45
      44 41 20 53 43
      53 53 45 52 44
      49 41 56 41 20
      45 4C 42 41 4C
      20 3A
7043 430
7043 431 20
7044 432 40
7045 433 15
7046 434
7046 435 10
7047 436 E0
7048 437 15
7049 438
7049 439 90
704A 440 C0
704B 441 15
704C 442
704C 443 88
704D 444 3F
704E 445 F4
704F 446
704F 447 00
7050 448 00
7051 449 00
7052 450 01
7053 451 FF
7054 452 FF
7055 453 FF
7056 454 FE
7057 455
7057 456 000A
7059 457 0802
705B 458 0200
705D 459
705D 460 000A
705F 461 0A02
7061 462
7061 463 000A
7063 464 0802
7065 465 5555
7067 466 0C0A
7069 467 A9A2
706B 468 A900
706D 469
706D 470 000A
706F 471 0802
7071 472 0800
7073 473 0801

```

```

      B 2
WCS_SIZE: .WORD
WCS_SIZER_ADDR: .WORD
                  .WORD
                  .WORD
                  .WORD
HEADER_B: .BYTE
          .ASCII

INC_WRO: .BYTE
         .BYTE
         .BYTE

SET_CPU_ATTN: .BYTE
              .BYTE
              .BYTE

CLR_CPU_ATTN: .BYTE
              .BYTE
              .BYTE

JMP_INST: .BYTE
          .BYTE
          .BYTE

ZERO_DAT: .BYTE
ONE_SHIFT: .BYTE
          .BYTE
          .BYTE

ONE_DAT: .BYTE
ZERO_SHIFT: .BYTE
          .BYTE
          .BYTE

TEST27_DAT: .WORD
            .WORD
            .WORD

TEST28_DAT: .WORD
            .WORD

TEST29_DAT: .WORD
            .WORD
            .WORD
            .WORD
            .WORD
            .WORD

TEST2A_DAT: .WORD
            .WORD
            .WORD
            .WORD

```

```

      Fiche 1 Frame B2 Sequence 14
0 ;SIZE OF WCS.
<^X003C> ;*
<^X0040> ;WCS ADDRESSES TO BE WRITTEN TO
<^X0044> ; DETERMINE IF WCS IS 16K,17K,
<^X0048> ; 18K,19K OR 20K.
<^X004C> ;*

32 ;32 CHARACTERS FOLLOWS...
<CR>'HIGHEST WCS ADDRESS AVAILABLE: ''

<^X20> ;INSTRUCTION TO INCREMENT
<^X40> ; WORKING REGISTER ZERO.
<^X15>

<^X10> ;INSTRUCTION TO SET THE
<^XE0> ; CPU ATTENTION SIGNAL.
<^X15>

<^X90> ;INSTRUCTION TO CLEAR THE
<^XC0> ; CPU ATTENTION SIGNAL.
<^X15>

<^X88> ;JUMP INSTRUCTION INTO WHICH
<^X3F> ; THE JUMP ADDRESS WILL BE
<^XF4> ; INSERTED DURING EXECUTION.

<^X00> ;*
<^X00> ;*
<^X00> ;*
<^X01> ;STANDARD DATA PATTERNS.
<^XFF> ;*
<^XFF> ;*
<^XFF> ;*
<^XFE> ;*

<^X000A> ;MASTER MODE DATA.
<^X0802> ;COUNTER MODE DATA.
<^X0200> ;COUNTER INITIAL VALUE.

<^X000A> ;MASTER MODE DATA.
<^X0A02> ;COUNTER MODE DATA.

<^X000A> ;MASTER MODE DATA.
<^X0802> ;COUNTER MODE DATA.
<^X5555> ;COUNTER RELOAD DATA.
<^X0C0A> ;SECOND MASTER MODE DATA.
<^XA9A2> ;SECOND COUNTER 1 MODE DATA.
<^XA900> ;SECOND COUNTER 2 MODE DATA.

<^X000A> ;MASTER MODE DATA.
<^X0802> ;COUNTER 1 MODE DATA.
<^X0800> ;COUNTER 2 MODE DATA.
<^X0801> ;COUNTER 4 MODE DATA.

```



```

ZZ-ENKBE-2.1 7000 4
7075 474 0800
7077 475 F0FF
7079 476 0200
707B 477
707B 478 0C0A
707D 479 A9A2
707F 480 A900
7081 481 0E02
7083 482
7083 483 0BC1'C3
7086 484
7086 485
7086 486
7086 487
7086 488
7086 489
7086 490
7086 491
7086 492
7086 493
7086 494
7086 495 02
7087 496 00000089
7089 497
7089 498 02
708A 499 0000008C
708C 500
708C 501 02
708D 502 0000008F
708F 503
708F 504
708F 505
708F 506
708F 507
708F 508
708F 509
708F 510
708F 511
708F 512 40D9 3A
7092 513 B7
7093 514 C8
7094 515 0000'CD
7097 516
7097 517 0000'3A
709A 518 0F
709B 519 00AD'D2
709E 520
709E 521 0000'2A
70A1 522 0007'22
70A4 523
70A4 524 0000'CD
70A7 525
70A7 526 0007'2A
70AA 527 0000'22
70AD 528
70AD 529 C9
70AE 530
70AE 531
70AE 532
70AE 533

```

```

C 2
Fiche 1 Frame C2 Sequence 15
;COUNTER 5 MODE DATA.
;COUNTER 4 INITIAL VALUE.
;COUNTER (2 AND 5) EXP DATA.
TEST2B_DAT: .WORD <^X0800>
               .WORD <^XF0FF>
               .WORD <^X0200>
               .WORD <^X0C0A>
               .WORD <^XA9A2>
               .WORD <^XA900>
               .WORD <^X0E02>
               ;MASTER MODE DATA.
               ;COUNTER 1 MODE DATA.
               ;COUNTER 2 MODE DATA.
               ;COUNTER 3 MODE DATA.
TEST2D_JUMP: JMP TEST2D_PAR_ERR ;INSTRUCTION USED TO
                                   ; RETURN FROM A PARITY
                                   ; ERROR.

;*****
; SHIFT VARIABLES - DO NOT SEPARATE PAIRS
;*****

TEMP_SHIFT: .BYTE 2 ;TEMPORARY STORAGE LOCATION FOR
TEMP_DAT: .BLKB 2 ; DATA FROM VARIOUS TESTS.

EXP_SHIFT: .BYTE 2 ;LOCATION OF EXPECTED DATA FROM
EXP_DAT: .BLKB 2 ; VARIOUS TESTS.

REC_SHIFT: .BYTE 2 ;LOCATION OF RECEIVED DATA FROM
REC_DAT: .BLKB 2 ; VARIOUS TESTS.

;*****
; CHECK_RD THIS ROUTINE CHECKS TO SEE IF DIAGNOSTIC IS UNDER RD CONTROL.
; IF IT IS, A CALL IS MADE TO GCVECT SO THAT THE BOOT CODE CAN
; KEEP TRACK OF ELAPSED TIME.
;*****

CHECK_RD: LDA MIPFLG ;GET STATE OF MODEM IN PROGRESS
           ORA A ;SET CONDITION CODES
           RZ ;RETURN IF ZERO
           CALL GCVECT ;ELSE GO TO BOOT CODE IF RD

           LDA CNTLC_Typed ;SEE IF CONTROL C TYPED (UNDER RD)
           RRC
           JNC 1$ ;IF NOT, BACK TO TEST

           LHLD WCS_ADDRESS ;ELSE SAVE THE PRESENT WCS ADDRESS.
           SHLD SAVE_WCS_ADD

           CALL SET_FOR_CONT ;RETURN TO MONITOR.

           LHLD SAVE_WCS_ADD ;RESTORE STATE.
           SHLD WCS_ADDRESS

1$: RET ;BACK TO TEST

;*****
; MASTER_RESET THIS ROUTINE PERFORMS AN INTERVAL TIMER MASTER RESET.

```

ZZ-ENKBE-2.1 7000 4

D 2

Fiche 1 Frame D2

Sequence 16

70AE 534
70AE 535
70AE 536
70AE 537
70AE 538 FF 3E
70B0 539 1D D3
70B2 540
70B2 541 C9
70B3 542
70B3 543
70B3 544
70B3 545
70B3 546
70B3 547
70B3 548
70B3 549
70B3 550
70B3 551 AF
70B4 552 000C'32
70B7 553
70B7 554 C9
70B8 555
70B8 556
70B8 557
70B8 558
70B8 559
70B8 560
70B8 561
70B8 562
70B8 563
70B8 564 00'0C'21
70BB 565 34
70BC 566
70BC 567 C9
70BD 568
70BD 569
70BD 570
70BD 571
70BD 572
70BD 573
70BD 574
70BD 575
70BD 576
70BD 577
70BD 578 17 3E
70BF 579 1D D3
70C1 580
70C1 581 0106'CD
70C4 582
70C4 583 C9
70C5 584
70C5 585
70C5 586
70C5 587
70C5 588
70C5 589
70C5 590
70C5 591
70C5 592
70C5 593

```

:*****
MASTER_RESET:  MVI    A,RESET_DATA  ;MOVE MASTER RESET DATA TO A.
                  OUT     ITCSR      ;PERFORM MASTER RESET.
                  RET

:*****
CLR.CG_PNT  THIS ROUTINE CLEARS THE COUNTER GROUP POINTER.
:*****

CLR.CG_PNT:  XRA     A              ;CLEAR A.
                  STA     CG_PNT     ;STORE IN CG_PNT.
                  RET

:*****
INC.CG_PNT  THIS ROUTINE INCREMENTS THE COUNTER GROUP POINTER.
:*****

INC.CG_PNT:  LXI     H,CG_PNT      ;POINT TO CG_PNT.
                  INR     M          ;INCREMENT CG_PNT.
                  RET

:*****
WRITE_MASTER THIS ROUTINE WRITE THE DATA, POINTED TO BY THE H,L PAIR,
              TO THE MASTER MODE REGISTER.
:*****

WRITE_MASTER: MVI     A,MASTER_MODE ;POINT TO THE MASTER MODE REG.
                  OUT     ITCSR
                  CALL    WRITE_ITDB ;WRITE THE DATA TO THE MASTER REG.
                  RET

:*****
WRITE_MODE  THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR,
            TO THE MODE REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
:*****

```



```

ZZ-ENKBE-2.1 7000 4
70C5 594 00 06
70C7 595
70C7 596
70C7 597 000C'3A
70CA 598 B0
70CB 599 1D D3
70CD 600
70CD 601 0106'CD
70D0 602
70D0 603 C9
70D1 604
70D1 605
70D1 606
70D1 607
70D1 608
70D1 609
70D1 610
70D1 611
70D1 612
70D1 613
70D1 614 08 06
70D3 615
70D3 616
70D3 617 000C'3A
70D6 618 B0
70D7 619 1D D3
70D9 620
70D9 621 0106'CD
70DC 622
70DC 623 C9
70DD 624
70DD 625
70DD 626
70DD 627
70DD 628
70DD 629
70DD 630
70DD 631
70DD 632
70DD 633
70DD 634 10 06
70DF 635
70DF 636
70DF 637 000C'3A
70E2 638 B0
70E3 639 1D D3
70E5 640
70E5 641 00FB'CD
70E8 642
70E8 643 C9
70E9 644
70E9 645
70E9 646
70E9 647
70E9 648
70E9 649
70E9 650
70E9 651
70E9 652
70E9 653

```

```

E 2
WRITE_MODE: MVI B,MODE_REG ;DATA FOR MODE REGISTER WITH COUNTER
; GROUP FIELD CLEARED.

LDA CG_PNT ;OR IN THE COUNTER GROUP POINTER.
ORA B
OUT ITCSR ;POINT TO THE CGX MODE REGISTER.

CALL WRITE_ITDB ;WRITE THE DATA TO THE MODE REGISTER.

RET

```

```

*****
WRITE_LOAD THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR,
TO THE LOAD REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
*****

```

```

WRITE_LOAD: MVI B,LOAD_REG ;DATA FOR LOAD REGISTER WITH COUNTER
; GROUP FIELD CLEARED.

LDA CG_PNT ;OR IN THE COUNTER GROUP POINTER.
ORA B
OUT ITCSR ;POINT TO THE CGX LOAD REGISTER.

CALL WRITE_ITDB ;WRITE THE DATA TO THE LOAD REGISTER.

RET

```

```

*****
READ_HOLD THIS ROUTINE READS THE DATA FROM THE HOLD REGISTER POINTED TO
BY THE COUNTER GROUP POINTER AND PUTS IT IN REC_DAT.
*****

```

```

READ_HOLD: MVI B,HOLD_REG ;DATA FOR HOLD REGISTER WITH COUNTER
; GROUP FIELD CLEARED.

LDA CG_PNT ;OR IN THE COUNTER GROUP POINTER.
ORA B
OUT ITCSR ;POINT TO THE CGX HOLD REGISTER.

CALL READ_ITDB ;READ THE DATA FROM THE HOLD REGISTER.

RET

```

```

*****
WRITE_ALARM THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR, TO
THE ALARM REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
*****

```

```

ZZ-ENKBE-2.1 7000 4
70E9 654 07 06
70EB 655
70EB 656 000C'3A
70EE 657 3D
70EF 658 00F4'CA
70F2 659
70F2 660 0F 06
70F4 661 78
70F5 662
70F5 663 1D D3
70F7 664
70F7 665 0106'CD
70FA 666
70FA 667 C9
70FB 668
70FB 669
70FB 670
70FB 671
70FB 672
70FB 673
70FB 674
70FB 675
70FB 676
70FB 677
70FB 678
70FB 679 1C DB
70FD 680 008E'32
7100 681
7100 682 1C DB
7102 683 008D'32
7105 684
7105 685 C9
7106 686
7106 687
7106 688
7106 689
7106 690
7106 691
7106 692
7106 693
7106 694
7106 695
7106 696
7106 697
7106 698 23
7107 699 7E
7108 700 1C D3
710A 701
710A 702 2B
710B 703 7E
710C 704 1C D3
710E 705
710E 706 C9
710F 707
710F 708
710F 709
710F 710
710F 711
710F 712
710F 713

```

```

F 2
WRITE_ALARM: MVI B,ALARM_REG1 ;DATA FOR ALARM REGISTER 1.
              LDA CG_PNT      ;ARE WE POINTING TO COUNTER GROUP 1.
              DCR A           ;SKIP NEXT INSTRUCTION IF SO.
              JZ 1$
              MVI B,ALARM_REG2 ;NO, LOAD DATA FOR ALARM REGISTER 2.
1$: MOV A,B      ;MOVE ALARM DATA INTO A.
              OUT ITCSR      ;POINT TO THE CGX ALARM REGISTER.
              CALL WRITE_ITDB ;WRITE THE DATA TO THE ALARM REG.
              RET

```

```

*****
: READ_ITDB THIS ROUTINE READS TWO BYTES OF DATA FROM THE INTERVAL TIMER
:           DATA BUS TO REC_DAT. THE DATA POINTER REGISTER SHOULD BE LOADED
:           PRIOR TO USING THIS ROUTINE.
*****

```

```

READ_ITDB: IN ITDB      ;READ THE LOW BYTE AND PUT IT IN
           STA REC_DAT+1 ; REC_DAT+1.
           IN ITDB      ;READ THE HIGH BYTE AND PUT IT IN
           STA REC_DAT  ; REC_DAT.
           RET

```

```

*****
: WRITE_ITDB THIS ROUTINE WRITES TWO BYTES OF DATA POINTED TO BY THE H,L
:            PAIR TO THE INTERVAL TIMER DATA BUS. THE DATA POINTER REGISTER
:            SHOULD BE LOADED PRIOR TO USING THIS ROUTINE.
*****

```

```

WRITE_ITDB: INX H      ;WRITE THE LOW BYTE TO THE
           MOV A,M      ; INTERVAL TIMER.
           OUT ITDB
           DCX H
           MOV A,M      ;WRITE THE HIGH BYTE.
           OUT ITDB
           RET

```

```

*****
: ARM_CNTR THIS ROUTINE ARMS THE COUNTER SPECIFIED BY CG_PNT.
*****

```


ZZ-ENKBE-2.1 7000 4

G 2

Fiche 1 Frame G2

Sequence 19

710F 714
710F 715
710F 716 000C'3A
7112 717 000D'32
7115 718
7115 719 AF
7116 720 3F
7117 721
7117 722 17
7118 723
7118 724 00'0D'21
711B 725 35
711C 726 0117'C2
711F 727
711F 728 20 F6
7121 729
7121 730 1D D3
7123 731
7123 732 C9
7124 733
7124 734
7124 735
7124 736
7124 737
7124 738
7124 739
7124 740
7124 741
7124 742 000C'3A
7127 743 000D'32
712A 744
712A 745 AF
712B 746 3F
712C 747
712C 748 17
712D 749
712D 750 00'0D'21
7130 751 35
7131 752 012C'C2
7134 753
7134 754 C0 F6
7136 755
7136 756 1D D3
7138 757
7138 758 C9
7139 759
7139 760
7139 761
7139 762
7139 763
7139 764
7139 765
7139 766
7139 767
7139 768
7139 769
7139 770 000C'3A
713C 771 000D'32
713F 772
713F 773 AF

```
ARM_CNTR:  LDA    CG_PNT      ;LOAD THE COUNTER GROUP POINTER IN A.
            STA    SHIFT_CNT  ;STORE A IN SHIFT_CNT.

            XRA    A          ;CLEAR A.
            CMC              ;SET THE CARRY.

1$:         RAL              ;SHIFT A THROUGH CARRY.

            LXI    H,SHIFT_CNT ;POINT TO SHIFT_CNT.
            DCR    M          ;DECREMENT SHIFT_CNT.
            JNZ    1$         ;IF ZERO, STOP SHIFTING.

            ORI    ARM_CNTR_DAT ;OR IN ARM COUNTER DATA.

            OUT    ITCSR       ;ARM COUNTER.

            RET
```

```
*****
:  DISARM_CNTR  THIS ROUTINE DISARMS THE COUNTER SPECIFIED BY CG_PNT.
*****
```

```
DISARM_CNTR: LDA    CG_PNT      ;LOAD THE COUNTER GROUP POINTER IN A.
              STA    SHIFT_CNT  ;STORE A IN SHIFT_CNT.

              XRA    A          ;CLEAR A.
              CMC              ;SET THE CARRY.

1$:         RAL              ;SHIFT A THROUGH CARRY.

              LXI    H,SHIFT_CNT ;POINT TO SHIFT_CNT.
              DCR    M          ;DECREMENT SHIFT_CNT.
              JNZ    1$         ;IF ZERO, STOP SHIFTING.

              ORI    DISARM_CNTR_DAT ;OR IN DISARM COUNTER DATA.

              OUT    ITCSR       ;DISARM COUNTER.

              RET
```

```
*****
:  LOAD_CNTR   THIS ROUTINE LOADS THE COUNTER OF THE COUNTER GROUP SPECIFIED BY
:              CG_PNT FROM THE SOURCE SPECIFIED BY THE COUNTER MODE REGISTER.
*****
```

```
LOAD_CNTR:  LDA    CG_PNT      ;MOVE THE COUNTER GROUP POINTER TO A.
            STA    SHIFT_CNT  ;STORE A IN SHIFT_CNT.

            XRA    A          ;CLEAR A.
```

ZZ-ENKBE-2.1 7000 4
 7140 774 3F
 7141 775
 7141 776 17
 7142 777
 7142 778 00'0D'21
 7145 779 35
 7146 780 0141'C2
 7149 781
 7149 782 40 F6
 714B 783
 714B 784 1D D3
 714D 785
 714D 786
 714D 787 C9
 714E 788
 714E 789
 714E 790
 714E 791
 714E 792
 714E 793
 714E 794
 714E 795
 714E 796
 714E 797
 714E 798 000C'3A
 7151 799 000D'32
 7154 800
 7154 801 AF
 7155 802 3F
 7156 803
 7156 804 17
 7157 805
 7157 806 00'0D'21
 715A 807 35
 715B 808 0156'C2
 715E 809
 715E 810 A0 F6
 7160 811
 7160 812 1D D3
 7162 813
 7162 814
 7162 815 C9
 7163 816
 7163 817
 7163 818
 7163 819
 7163 820
 7163 821
 7163 822
 7163 823
 7163 824
 7163 825
 7163 826 00'0C'21
 7166 827 E8 3E
 7168 828 B6
 7169 829
 7169 830 1D D3
 716B 831
 716B 832 C9
 716C 833

H 2

Fiche 1 Frame H2

Sequence 20

1\$:

```

CMC          ;SET THE CARRY.
RAL          ;SHIFT A THROUGH CARRY.
LXI          H,SHIFT_CNT ;POINT TO SHIFT CNT.
DCR          M          ;DECREMENT SHIFT CNT.
JNZ          1$         ;IF ZERO, STOP SHIFTING.
ORI          LOAD_CNTR_DAT ;OR IN LOAD COUNTER DATA.
OUT          ITCSR       ;LOAD COUNTER FROM REGISTER SPECIFIED
                        ; BY COUNTER MODE REGISTER.
RET
  
```

```

*****
:  SAVE_CNTR  THIS ROUTINE SAVES THE COUNTER SPECIFIED BY CG_PNT IN THE
:              CORRESPONDING HOLD REGISTER.
*****
  
```

```

SAVE_CNTR:   LDA      CG_PNT      ;MOVE THE COUNTER GROUP POINTER TO A.
              STA      SHIFT_CNT   ;STORE A IN SHIFT_CNT.
              XRA      A           ;CLEAR A.
              CMC          ;SET THE CARRY.
1$:          RAL          ;SHIFT A THROUGH CARRY.
              LXI          H,SHIFT_CNT ;POINT TO SHIFT CNT.
              DCR          M          ;DECREMENT SHIFT CNT.
              JNZ          1$         ;IF ZERO, STOP SHIFTING.
              ORI          SAVE_CNTR_DAT ;OR IN SAVE COUNTER DATA.
              OUT          ITCSR       ;SAVE COUNTER IN CORRESPONDING
                        ; SAVE REGISTER.
              RET
  
```

```

*****
:  SET_OUTPUT THIS ROUTINE SET THE OUTPUT BIT OF THE COUNTER GROUP SPECIFIED
:              BY CG_PNT.
*****
  
```

```

SET_OUTPUT:  LXI          H,CG_PNT ;POINT TO THE COUNTER GROUP.
              MVI          A,SET_OUTPUT_DAT ;OR IN THE SET OUTPUT DATA.
              ORA          M
              OUT          ITCSR       ;SET THE OUTPUT BIT.
              RET
  
```


22-ENKBE-2.1 7000 4

I 2

Fiche 1 Frame I2

Sequence 21

716C 834
716C 835
716C 836
716C 837
716C 838
716C 839
716C 840
716C 841
716C 842
716C 843 00'0C'21
716F 844 E0 3E
7171 845 B6
7172 846
7172 847 1D D3
7174 848
7174 849 C9
7175 850
7175 851
7175 852
7175 853
7175 854
7175 855
7175 856
7175 857
7175 858
7175 859 00'0C'21
7178 860 F0 3E
717A 861 B6
717B 862
717B 863 1D D3
717D 864
717D 865 C9
717E 866
717E 867
717E 868
717E 869
717E 870
717E 871
717E 872
717E 873
717E 874
717E 875
717E 876 0010'32
7181 877
7181 878 C5 46 00'00'21
77 64 3E
7189 879 00
718A 880 35 00'00'21
70 C1 0189'C2
7193 881
7193 882 0010'3A
7196 883 3D
7197 884 0010'32
719A 885 017E'C2
719D 886
719D 887 C9
719E 888
719E 889
719E 890
719E 891

*
*

CLR_OUTPUT THIS ROUTINE CLEARS THE OUTPUT BIT OF THE COUNTER GROUP
SPECIFIED BY CG_PNT.

CLR_OUTPUT: LXI H,CG_PNT ;POINT TO THE COUNTER GROUP.
MVI A,CLR_OUTPUT_DAT ;OR IN THE CLEAR OUTPUT DATA.
ORA M
OUT ITCR ;CLEAR THE OUTPUT BIT.
RET

INC_CNTR THIS ROUTINE INCREMENTS THE COUNTER SPECIFIED BY CG_PNT.

INC_CNTR: LXI H,CG_PNT ;POINT TO THE COUNTER GROUP.
MVI A,INC_CNTR_DAT ;OR IN THE INCREMENT COUNTER
ORA M ; DATA.
OUT ITCR ;INCREMENT THE COUNTER.
RET

WAIT_LOOP THIS ROUTINE WAITS A PERIOD OF TIME SPECIFIED BY THE COUNT
IN THE ACCUMULATOR.

WAIT_LOOP: STA WAIT_CNT ;STORE THE WAIT COUNT IN A.
DO 100 ;INCREASE WAITING TIME.
NOP
ENDDO
LDA WAIT_CNT ;DECREMENT THE WAIT COUNT.
DCR A
STA WAIT_CNT
JNZ WAIT_LOOP
RET ;RETURN WHEN THE WAIT COUNT IS UP.

```
SET_TR_LB:      XRA      A      ;CLEAR REMOTE USART CSR BEFORE
                OUT      TRCR   ;CHANGING TO LOOPBACK MODE
```

[illegible]

ZZ-ENKBE-2.1 7000 4
71CC 952 B3 3E
71CE 953 43 D3
71D0 954
71D0 955 41 DB
71D2 956 1F
71D3 957 01D0'D2
71D6 958
71D6 959 C9
71D7 960
71D7 961
71D7 962
71D7 963
71D7 964
71D7 965
71D7 966
71D7 967
71D7 968
71D7 969 AF
71D8 970 4B D3
71DA 971
71DA 972 35 3E
71DC 973 4B D3
71DE 974
71DE 975 C9
71DF 976
71DF 977
71DF 978
71DF 979
71DF 980
71DF 981
71DF 982
71DF 983
71DF 984
71DF 985 AF
71E0 986 47 D3
71E2 987
71E2 988 15 3E
71E4 989 47 D3
71E6 990
71E6 991 C9
71E7 992
71E7 993
71E7 994
71E7 995
71E7 996
71E7 997
71E7 998
71E7 999
71E7 1000
71E7 1001 AF
71E8 1002 43 D3
71EA 1003
71EA 1004 15 3E
71EC 1005 43 D3
71EE 1006
71EE 1007 C9
71EF 1008
71EF 1009
71EF 1010
71EF 1011

K 2

Fiche 1 Frame K2 Sequence 23

MVI A,UA_LOOPBACK ;MOVE LOOPBACK DATA TO A.
OUT TRCR ;SEND IT TO THE REMOTE USART.

1\$: IN TRSR ;WAIT FOR A TRANSMIT READY SIGNAL.
RAR
JNC 1\$

RET

: RESTORE_TT THIS ROUTINE RESTORES THE TERMINAL USART TO NORMAL MODE.
:*****

RESTORE_TT: XRA A ;CLEAR TERMINAL USART CSR BEFORE
OUT TTCR ;CHANGING TO NORMAL MODE.

MVI A,UATT_RESTORE ;MOVE USART RESTORE DATA TO A.
OUT TTCR ;SEND IT TO THE TERMINAL USART.

RET

: RESTORE_TU THIS ROUTINE RESTORES THE TU58 USART TO NORMAL MODE.
:*****

RESTORE_TU: XRA A ;CLEAR TU USART CSR BEFORE
OUT TUCR ;CHANGING TO NORMAL MODE.

MVI A,UA_RESTORE ;MOVE USART RESTORE DATA TO A.
OUT TUCR ;SEND IT TO THE TU58 USART.

RET

: RESTORE_TR THIS ROUTINE RESTORES THE REMOTE USART TO NORMAL MODE.
:*****

RESTORE_TR: XRA A ;CLEAR REMOTE USART CSR BEFORE
OUT TRCR ;CHANGING TO NORMAL MODE.

MVI A,UA_RESTORE ;MOVE USART RESTORE DATA TO A.
OUT TRCR ;SEND IT TO THE REMOTE USART.

RET

:

22-ENKBE-2.1 7000 4

L 2

Fiche 1 Frame L2

Sequence 24

71EF 1012
71EF 1013
71EF 1014
71EF 1015
71EF 1016
71EF 1017
71EF 1018
71EF 1019
71EF 1020 000A'32 AF
71F3 1021 000B'32
71F6 1022
71F6 1023 40D9 3A
71F9 1024 0F
71FA 1025 0205'D2
71FD 1026
71FD 1027
71FD 1028 000A'32 3C AF
7202 1029 000B'32
7205 1030
7205 1031 0000'3A
7208 1032 0F
7209 1033 0211'D2
720C 1034
720C 1035 000B'32 3C AF
7211 1036
7211 1037
7211 1038 C9
7212 1039
7212 1040
7212 1041
7212 1042
7212 1043
7212 1044
7212 1045
7212 1046
7212 1047
7212 1048
7212 1049
7212 1050
7212 1051 0018'2A
7215 1052 0016'22
7218 1053
7218 1054 00'18'21
721B 1055 0014'22
721E 1056
721E 1057 C5 46 00'00'21
77 05 3E
7226 1058
7226 1059 0014'2A
7229 1060 00'00'11
722C 1061 02 0E
722E 1062 0000'CD
7231 1063
7231 1064 00'4F'21
7234 1065 00'00'11
7237 1066 0000'CD
723A 1067
723A 1068 0000'CD
723D 1069
723D 1070 0000'CD

*

*

*

*

*

*

*

*

*

*

*

*

*

*

```

: CHECK_REMOTE  THIS ROUTINE CHECKS TO SEE IF DIAGNOSTICS ARE BEING RUN
:               REMOTELY OR UNDER APT. IF SO, USART3_FLG IS SET TO INDICATE
:               THAT USART 3 SHOULD NOT BE TESTED. USART 3 IS TESTED ON
:               THE FIRST PASS UNDER APT.
:*****

```

```

CHECK_REMOTE:  CLEAR  SKIP-UA1  ;CLEAR THE USART1 FLAG.
               STA    SKIP-UA3  ;CLEAR THE USART3 FLAG.

               LDA    MIPFLG    ;READ MODEM IN PROGRESS FLAG.
               RRC
               JNC     1$        ;DON'T SET FLAG IF NOT UNDER
                                ;REMOTE DIAGNOSTICS.

               SET     SKIP-UA1  ;SET THE USART FLAGS. USARTS 1 AND 3
               STA    SKIP-UA3  ; WILL NOT BE TESTED.

1$:            LDA    APT        ;CHECK THE APT PRESENT FLAG.
               RRC
               JNC     2$        ;DON'T SET FLAG IF APT NOT PRESENT.

               SET     SKIP-UA3  ;SET THE USART3 FLAG. USART 3 WILL
                                ; NOT BE TESTED.

2$:            RET

```

```

:*****
: WCS_SIZER  THIS ROUTINE SIZES THE WCS BY WRITING DATA PATTERNS AT 17K,
:            18K, 19K AND 20K AND THEN READING THESE PATTERNS BACK. IF THE
:            PATTERN RECEIVED IS THE SAME AS THE ONE SENT OUT THEN WCS IS
:            ASSUMED TO EXIST AT THAT LOCATION.
:*****

```

```

WCS_SIZER:    LHLD   WCS_SIZER_ADDR ;SET INITIAL WCS SIZE TO 16K.
               SHLD  WCS_SIZE

               LXI   H,WCS_SIZER_ADDR ;SIZE WCS AT 16K.
               SHLD  WCS_ADDR_PNT

               DO     5

               LHLD  WCS_ADDR_PNT  ;PREPARE TO WRITE TO WCS.
               LXI   D,WCS_ADDRESS
               MVI   C,2
               CALL  MOVER

               LXI   H,ZERO DAT    ;LOAD DATA TO BE WRITTEN TO WCS
               LXI   D,WCS_VALUE  ; INTO WCS_VALUE.
               CALL  MOVER_3

               CALL  WCS_WRITE    ;WRITE TO WCS.

               CALL  WCS_READ     ;READ FROM WCS.

```


ZZ-ENKBE-2.1 7000 4

7240 1071
7240 1072 00'00'21
03 0E 00'53'11
0251'C2 0000'CD

724E 1073
724E 1074 027A'C3

7251 1075
7251 1076

7251 1077
7251 1078 0C8F'CD

7254 1079
7254 1080 00'4F'21

7257 1081 00'00'11
725A 1082 0000'CD

725D 1083
725D 1084 0000'CD

7260 1085
7260 1086 0000'CD

7263 1087
7263 1088 00'00'21

03 0E 00'53'11
0274'C2 0000'CD

7271 1089
7271 1090 027A'C3

7274 1091
7274 1092

7274 1093
7274 1094 0000'2A

7277 1095 0016'22
727A 1096

727A 1097 0014'2A
727D 1098 23

727E 1099 23
727F 1100 0014'22

7282 1101
7282 1102 008F'CD

7285 1103
7285 1104 35 00'00'21

70 C1 0226'C2

728E 1105
728E 1106 0016'3A

7291 1107 03 F6
7293 1108 0016'32

7296 1109
7296 1110 0017'3A

7299 1111 FF F6
729B 1112 0017'32

729E 1113
729E 1114 C9

729F 1115
729F 1116

729F 1117
729F 1118

729F 1119
729F 1120

729F 1121
729F 1122

729F 1123
729F 1124

729F 1125 00'22'21

M 2

Fiche 1 Frame M2

Sequence 25

IFEQ WCS_VALUE,ONE_DAT,3 ;IF RECEIVED DATA IS ALL ONES...

JMP 1\$;...LEAVE WCS_SIZE AS IS.

ENDIF

CALL CHECK_RD ;CALL BOOT CODE IF RD IS IN CONTROL

LXI H,ZERO_DAT ;LOAD DATA TO BE WRITTEN TO WCS
LXI D,WCS_VALUE ; INTO WCS_VALUE.

CALL MOVER_3

CALL WCS_WRITE ;WRITE TO WCS.

CALL WCS_READ ;READ FROM WCS.

IFEQ WCS_VALUE,ONE_DAT,3 ;IF RECEIVED DATA IS ALL ONES...

JMP 1\$;...LEAVE WCS_SIZE AS IS.

ENDIF

LHLD WCS_ADDRESS ;PRESENT WCS LOCATION EXISTS. SET
SHLD WCS_SIZE ; WCS SIZE TO THIS LOCATION.

1\$: LHLD WCS_ADDR_PNT ;TRY THE NEXT HIGHEST WCS LOCATION.

INX H

INX H

SHLD WCS_ADDR_PNT

CALL CHECK_RD ;CALL BOOT CODE IF RD IS IN CONTROL

ENDDO

LDA WCS_SIZE ;SET BITS 8 AND 9 OF WCS_SIZE TO
ORI 3 ; DISPLAY THE HIGHEST WCS ADDRESS
STA WCS_SIZE ; POSSIBLE.

LDA WCS_SIZE+1 ;SET BITS 0-7 OF WCS_SIZE.

ORI <^XFF>

STA WCS_SIZE+1

RET

WCS_SIZE_P THIS ROUTINE IS USED AFTER THE WCS SIZER ROUTINE TO
PRINT THE HIGHEST WCS ADDRESS AVAILCABLE.

WCS_SIZE_P: LXI H,HEADER_B ;PRINT THE WCS SIZE MESSAGE.

```

ZZ-ENKBE-2.1 7000 4
72A2 1126 0000'CD
72A5 1127
72A5 1128 00'16'21
72A8 1129 00'00'11
72AB 1130 02 0E
72AD 1131 0000'CD
72B0 1132 02 0E
72B2 1133 0000'CD
72B5 1134
72B5 1135 C9
72B6 1136
72B6 1137
72B6 1138
72B6 1139
72B6 1140
72B6 1141
72B6 1142
72B6 1143
72B6 1144
72B6 1145
72B6 1146 00'8E'11
72B9 1147 02BF'CD
72BC 1148
72BC 1149 00'8D'11
72BF 1150
72BF 1151 EC D3
72C1 1152 80 DB
72C3 1153 12
72C4 1154 00'00'21
72C7 1155
72C7 1156 7E
72C8 1157 000D'32
72CB 1158
72CB 1159 23
72CC 1160 0012'22
72CF 1161
72CF 1162 008F'CD
72D2 1163
72D2 1164 0012'2A
72D5 1165 00'00'11
72D8 1166 0000'CD
72DB 1167 0012'22
72DE 1168
72DE 1169 0000'CD
72E1 1170
72E1 1171 23 D3
72E3 1172 22 D3
72E5 1173
72E5 1174 00'0D'21
72E8 1175 35
72E9 1176 02CF'C2
72EC 1177
72EC 1178 00'00'21
72EF 1179 00'00'11
72F2 1180 0000'CD
72F5 1181
72F5 1182 0000'C3
72F8 1183
72F8 1184
72F8 1185

```

N 2

Fiche 1 Frame N2

Sequence 26

```

CALL PRINT_STRING
LXI H,WCS_SIZE ;PRINT THE HIGHEST WCS ADDRESS.
LXI D,HEX_BUF
MVI C,2
CALL MOVER
MVI C,2
CALL PRINT_HEX
RET

```

```

*****
: READ_DATA_16 THIS ROUTINE READS 2 BYTES OF DATA AND STORES IT IN REC_DAT
: FOR TEST 2D.
*****

```

READ_DATA_16:

```

LXI D,REC_DAT+1 ;DE POINTS TO DATA AREA
CALL 1$ ;READ LOW BYTE AND STORE

LXI D,REC_DAT ;DE POINTS TO DATA AREA
; FOR NEXT BYTE
1$: OUT YBUSRD
IN READ ;READ MOST SIG. BYTE
STAX D ;STORE IN DATA BUFFER
LXI H,X_SHIFT_R ;ADDR OF CSR INSTS TO EXECUTE

MOV A,M ;GET COUNT
STA SHIFT_CNT ;AND SAVE

INX H
SHLD TEMP_WORD

2$: CALL CHECK_RD ;CALL BOOT CODE IF RD IS IN CONTROL

LHLD TEMP_WORD
LXI D,CSR_VALUE ;MOVE 3 BYTES OF INST TO CSR_VALUE
CALL MOVER_3
SHLD TEMP_WORD

CALL LOAD_CSR ;PUT INST IN CSR

OUT SETSS
OUT CLRSS ;SINGLE STEP CPU

LXI H,SHIFT_CNT
DCR M ;DECREMENT THE COUNT
JNZ 2$ ;LOOP UNTIL ZERO

LXI H,X MOV_WRRW ;LEAVE MOV INST IN CSR TO
LXI D,CSR_VALUE ;ENABLE Y-BUS
CALL MOVER_3

JMP LOAD_CSR ;LOAD MOV

```


22-ENKBE-2.1 7000 4
72F8 1186
72F8 1187
72F8 1188
72F8 1189
72F8 1190
72F8 1191
72F8 1192
72F8 1193
72F8 1194
72F8 1195 0300'CD
72FB 1196
72FB 1197 23 D3
72FD 1198 22 D3
72FF 1199
72FF 1200 C9
7300 1201
7300 1202
7300 1203
7300 1204
7300 1205
7300 1206
7300 1207
7300 1208
7300 1209
7300 1210
7300 1211 00'00'11
7303 1212 0000'CD
7306 1213
7306 1214
7306 1215 A2 D3
7308 1216
7308 1217 00'00'21
730B 1218 0000'22
730E 1219
730E 1220 C5 46 00'00'21
77 18 3E
7316 1221
7316 1222 38 D3
7318 1223
7318 1224 87 DB
731A 1225 1F
731B 1226 0000'CD
731E 1227
731E 1228 0323'D2
7321 1229
7321 1230 39 D3
7323 1231
7323 1232 25 D3
7325 1233 24 D3
7327 1234
7327 1235 35 00'00'21
70 C1 0316'C2
7330 1236
7330 1237 A3 D3
7332 1238
7332 1239 C9
7333 1240
7333 1241
7333 1242
7333 1243

B 3

Fiche 1 Frame B3

Sequence 27

```
*****  
: EXEC_INST THIS ROUTINE EXECUTES AN INSTRUCTION POINTED TO BY THE H,L  
: PAIR.  
:*****
```

```
EXEC_INST:    CALL    LD_CSR_INST    ;LOAD INSTRUCTION INTO THE CSR.  
              OUT     SETSS          ;CLOCK THE CPU.  
              OUT     CLRSS  
              RET
```

```
*****  
: LD_CSR_INST THIS ROUTINE LOADS THE CSR WITH AN INSTRUCTION, WHICH IS  
: POINTED TO BY THE CONTENTS OF THE H,L PAIR.  
:*****
```

```
LD_CSR_INST:  LXI     D,CSR_VALUE    ;MOVE INSTRUCTION POINTED TO BY THE  
              CALL    MOVER_3        ; H,L PAIR INTO CSR VALUE.  
  
              OUT     VSHIFT         ;SET V-BUS TO SHIFTING MODE.  
  
              LXI     H,CSR_SHIFT    ;PREPARE TO SHIFT DATA INTO CSR.  
              SHLD    SHIFT_PNT  
  
              DO      24  
  
              OUT     CLRCSR         ;CLEAR THE CSR SERIAL INPUT.  
  
              IN      CSR23          ;READ THE CSR'S MSB.  
              RAR                     ;ROTATE IT INTO THE CARRY.  
              CALL    SHIFTER        ;SHIFT THE DATA THROUGH CSR_VALUE.  
  
              JNC     1$             ;IF THE CARRY IS SET...  
  
              OUT     SETCSR         ;SET THE CSR SERIAL INPUT.  
  
              OUT     SETCSK         ;CLOCK THE CSR SERIAL LOAD.  
              OUT     CLRCSK  
  
              ENDDO  
  
              OUT     VNORML         ;SET V-BUS TO NORMAL MODE.  
  
              RET
```

```
*****  
:  
:
```

```

ZZ-ENKBE-2.1 7000 4
7333 1244
7333 1245
7333 1246
7333 1247
7333 1248
7333 1249
7333 1250
7333 1251 88 3E
7335 1252 004C'32
7338 1253
7338 1254 3F 3E
733A 1255 004D'32
733D 1256
733D 1257 F4 3E
733F 1258 004E'32
7342 1259
7342 1260 0016'3A
7345 1261 0F
7346 1262 0F
7347 1263 0F
7348 1264 0F
7349 1265 03 E6
734B 1266
734B 1267 00'4C'21
734E 1268 B6
734F 1269 77
7350 1270
7350 1271 23
7351 1272
7351 1273 0016'3A
7354 1274 0F
7355 1275 0F
7356 1276 0F
7357 1277 0F
7358 1278 C0 E6
735A 1279
735A 1280 B6
735B 1281 77
735C 1282
735C 1283 0016'3A
735F 1284 40 E6
7361 1285 C8
7362 1286
7362 1287 00'01'21
7365 1288 02F8'CD
7368 1289
7368 1290 C9
7369 1291
7369 1292
7369 1293
7369 1294
7369 1295
7369 1296
7369 1297
7369 1298
7369 1299
7369 1300
7369 1301 FF FF 21
736C 1302 0000'22
736F 1303 02 0E 00'02'21

```

C 3

Fiche 1 Frame C3

Sequence 28

```

: OR_JMP_ADD THIS ROUTINE ORS IN THE WCS SIZE VALUE INTO THE JUMP ADDRESS
: FIELD OF THE JUMP INSTRUCTION. THE WCS PAGE BIT IS ALSO SET
: IF THE VALUE OF WCS SIZE EXCEEDS 16K.
:
: *****

```

```

OR_JMP_ADD: MVI A,<^X88> ;INITIALIZE THE JUMP INST.
            STA JMP_INST
            MVI A,<^X3F> ;*
            STA JMP_INST+1
            MVI A,<^XF4> ;*
            STA JMP_INST+2
            LDA WCS_SIZE ;LOAD HIGH BYTE OF WCS SIZE INTO A.
            RRC ;SHIFT RIGHT FOUR TIMES.
            RRC
            RRC
            RRC
            ANI <^X03> ;CLEAR BITS 2-7 OF A.
            LXI H,JMP_INST ;POINT TO HIGH BYTE OF JMP_INST.
            ORA M ;OR IN BITS 12 AND 13 OF WCS_SIZE.
            MOV M,A ;RETURN HIGH BYTE OF INSTRUCTION.
            INX H ;POINT TO MIDDLE BYTE OF JMP_INST.
            LDA WCS_SIZE ;LOAD HIGH BYTE OF WCS_SIZE INTO A.
            RRC ;SHIFT RIGHT FOUR TIMES.
            RRC
            RRC
            RRC
            ANI <^XC0> ;CLEAR BITS 0-5 OF A.
            ORA M ;OR IN BITS 10 AND 11 OF WCS_SIZE.
            MOV M,A ;RETURN MIDDLE BYTE OF INSTRUCTION.
            LDA WCS_SIZE ;CHECK FOR WCS_SIZE BIT 15 SET.
            ANI <^X40>
            RZ
            LXI H,X_SET_PAGE+1 ;SET THE PAGE BIT IF WCS EXCEEDS 16K.
            CALL EXEC_INST
            RET

```

```

: *****
: REPORT_ERROR THIS ROUTINE SPECIFIES WCS MODULE AND CALLS THE MONITOR
: TO REPORT AN ERROR.
:
: *****

```

```

REPORT_ERROR_2: LXI H,<^XFFFF> ;SECOND TWO BYTES ARE OF INTEREST.
                SHLD CON_MSK_DAT
                ZERO CON_MSK_DAT+2,2

```


0D 23 77 AF
0375'C2

737B 1304
737B 1305 00'00'21
737E 1306
737E 1307 00'00'11
7381 1308 05 0E
7383 1309 0000'CD
7386 1310
7386 1311 0000'CD
7389 1312
7389 1313 C9
738A 1314
738A 1315
738A 1316
738A 1317
738A 1318
738A 1319
738A 1320
738A 1321
738A 1322
738A 1323
738A 1324 0000'32 3C AF
738F 1325
738F 1326 03 0E 00'00'21
0D 23 77 AF
0395'C2
739B 1327 000C'3A
739E 1328 0003'32
73A1 1329
73A1 1330 FF FF 21
73A4 1331 0000'22
73A7 1332 0002'22
73AA 1333
73AA 1334 037B'C3
73AD 1335
73AD 1336
73AD 1337
73AD 1338
73AD 1339
73AD 1340
73AD 1341
73AD 1342
73AD 1343
73AD 1344 01 3E
73AF 1345 0000'32
73B2 1346
73B2 1347 008A'3A
73B5 1348 0003'32
73B8 1349
73B8 1350 008D'3A
73BB 1351 0003'32
73BE 1352
73BE 1353 0000'32 3C AF
73C3 1354
73C3 1355 FF FF 21
73C6 1356 0000'22
73C9 1357 C1 FF 21
73CC 1358 0002'22
73CF 1359

REPORT_ERROR: LXI H,MWCS_A ;WCS MODULE BEING TESTED.
LXI D,CON_MOD_DAT ;LOAD THE MODULE DATA POINTED TO BY
MVI C,5 ; THE H,L PAIR INTO CON_MOD_DAT.
CALL MOVER
CALL CON_ERROR ; PRINT ERROR MESSAGE.
RET

: TEST27_ERROR THIS ROUTINE IS USED BY TEST 27 TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
: *****

TEST27_ERROR: SET NA_EXP_REC ;NO EXPECTED OR RECEIVED DATA.
ZERO CON_ADD_DAT,3 ;ADDITIONAL DATA CONTAINS COUNTER
LDA CG_PNT ; GROUP POINTER.
STA CON_ADD_DAT+3
LXI H,<^XFFFF> ;NO DATA.
SHLD CON_MSK_DAT
SHLD CON_MSK_DAT+2
JMP REPORT_ERROR ;REPORT THE ERROR

: TEST28_ERROR THIS ROUTINE IS USED BY TEST 28 TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
: *****

TEST28_ERROR: MVI A,1 ;ERROR NUMBER 1.
STA CON_ERR_NUM
LDA EXP_DAT ;MOVE EXPECTED DATA TO CON_EXP_DAT.
STA CON_EXP_DAT+3
LDA REC_DAT ;MOVE RECEIVED DATA TO CON_REC_DAT.
STA CON_REC_DAT+3
SET NA_OTHER ;NO ADDITIONAL DATA.
LXI H,<^XFFFF> ;DATA OF INTEREST IS BITS 3-5.
SHLD CON_MSK_DAT
LXI H,<^XC1FF>
SHLD CON_MSK_DAT+2

```

ZZ-ENKBE-2.1 7000 4
73CF 1360 037B'C3
73D2 1361
73D2 1362
73D2 1363
73D2 1364
73D2 1365
73D2 1366
73D2 1367
73D2 1368
73D2 1369
73D2 1370
73D2 1371 008A'2A
73D5 1372 0002'22
73D8 1373
73D8 1374 008D'2A
73DB 1375 0002'22
73DE 1376
73DE 1377 03 0E 00'00'21
              0D 23 77 AF
              03E4'C2
73EA 1378 000C'3A
73ED 1379 0003'32
73F0 1380
73F0 1381 0369'C3
73F3 1382
73F3 1383
73F3 1384
73F3 1385
73F3 1386
73F3 1387
73F3 1388
73F3 1389
73F3 1390
73F3 1391
73F3 1392 008A'2A
73F6 1393 0002'22
73F9 1394
73F9 1395 008D'2A
73FC 1396 0002'22
73FF 1397
73FF 1398 0000'32 3C AF
7404 1399
7404 1400 0369'C3
7407 1401
7407 1402
7407 1403
7407 1404
7407 1405
7407 1406
7407 1407
7407 1408
7407 1409
7407 1410
7407 1411 0000'32 3C AF
740C 1412
740C 1413 0000'32 3C AF
7411 1414
7411 1415 FF FF 21
7414 1416 0000'22
7417 1417 0002'22

```

E 3

JMP REPORT_ERROR ;REPORT THE ERROR

Fiche 1 Frame E3

Sequence 30

```

*****
: TEST29_ERROR THIS ROUTINE IS USED BY TEST 29 TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
*****

```

```

TEST29_ERROR:  LHLD  EXP_DAT      ;MOVE EXPECTED DATA TO CON_EXP_DAT.
                SHLD  CON_EXP_DAT+2
                LHLD  REC_DAT      ;MOVE RECEIVED DATA TO CON_REC_DAT.
                SHLD  CON_REC_DAT+2
                ZERO  CON_ADD_DAT,3 ;ADDITONAL DATA CONTAINS COUNTER
                LDA   CG_PNT        ; GROUP POINTER.
                STA   CON_ADD_DAT+3
                JMP   REPORT_ERROR_2 ;REPORT THE ERROR (2 BYTES OF INTEREST)

```

```

*****
: TEST2A_ERROR THIS ROUTINE IS USED BY TEST 2A TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
*****

```

```

TEST2A_ERROR:  LHLD  EXP_DAT      ;MOVE EXPECTED DATA TO CON_EXP_DAT.
                SHLD  CON_EXP_DAT+2
                LHLD  REC_DAT      ;MOVE RECEIVED DATA TO CON_REC_DAT.
                SHLD  CON_REC_DAT+2
                SET   NA_OTHER      ;NO ADDITIONAL DATA.
                JMP   REPORT_ERROR_2 ;REPORT THE ERROR (2 BYTES OF INTEREST)

```

```

*****
: TEST2A_2_ERROR THIS ROUTINE IS USED BY TEST 2A ERROR 02 TO DEFINE THE
: ERROR VARIABLES IN PREPARATION FOR USING THE CON_ERROR
: ROUTINE.
*****

```

```

TEST2A_2_ERROR: SET   NA_EXP_REC    ;NO EXPECTED OR RECEIVED DATA.
                SET   NA_OTHER      ;NO ADDITIONAL DATA.
                LXI   H,<^XFFFF>    ;NO DATA.
                SHLD  CON_MSK_DAT
                SHLD  CON_MSK_DAT+2

```



```
TEST2D_ERROR:  LHLD    EXP_DAT      ;MOVE EXPECTED DATA TO CON_EXP_DAT.
                SHLD    CON_EXP_DAT+2
```

```

ZZ-ENKBE-2.1 7000 4
7455 1476
7455 1477 008D'2A
7458 1478 0002'22
745B 1479
745B 1480 0369'C3
745E 1481
745E 1482
745E 1483
745E 1484
745E 1485
745E 1486
745E 1487
745E 1488
745E 1489
745E 1490
745E 1491
745E 1492
745E 1493
745E 1494
745E 1495
745E 1496
745E 1497
745E 1498
745E 1499
745E 1500
745E 1501
745E 1502
745E 1503
745E 1504
745E 1505
745E 1506
745E 1507
745E 1508
745E 1509
745E 1510
745E 1511
745E 1512
745E 1513
745E 1514
745E 1515
745E 1516
745E 1517
745E 1518
745E 1519
745E 1520
745E 1521
745E 1522
745E 1523
745E 1524
745E 1525
745E 1526
745E 1527
745E 1528
745E 1529
745E 1530
745E 1531
745E 1532 OF 40D9 3A
27 3E 0551'DA
0000'3A 0000'CD
C9 0472'D2 OF

```

G 3

Fiche 1 Frame G3

Sequence 32

```

LHLD REC_DAT ;MOVE RECEIVED DATA TO CON REC DAT.
SHLD CON_REC_DAT+2
JMP REPORT_ERROR_2 ;REPORT THE ERROR (2 BYTES OF INTEREST)

```

TEST27

COUNTER RUN TEST

THIS TEST LOADS EACH OF THE COUNTERS WITH AN INITIAL VALUE AND THEN TURNS THE COUNTER ON. THE COUNTER VALUE IS CHECKED VIA THE HOLD REGISTER TO DETERMINE IF THE COUNTER HAS REACHED ZERO IN AN APPROPRIATE AMOUNT OF TIME. THE SIGNAL USED TO COUNT ON IS THE 5 MHZ SIGNAL AT SRC 2 PIN.

TEST STEPS:

- 1) PERFORM MASTER RESET.
- 2) LOAD MASTER MODE REGISTER WITH 0A00, DISABLING ALARM COMPARES AS WELL AS VARIOUS OTHER FUNCTIONS.
- 3) LOAD COUNTER MODE REGISTERS WITH 0209 IN PREPARATION FOR COUNTING UP ONCE IN BINARY AND RELOADING FROM THE LOAD REGISTER.
- 4) LOAD COUNTER WITH INITIAL VALUE.
- 5) LOAD LOAD REGISTER WITH ZERO
- 6) START COUNTER.
- 7) ENTER SHORT WAIT LOOP, THEN CHECK COUNTER'S CONTENTS.
- 8) ENTER LONG WAIT LOOP, THEN CHECK COUNTER'S CONTENTS.
- 9) PRINT AN ERROR IF COUNTER HAS REACHED TC IN TOO SHORT OR LONG AN INTERVAL.
- 10) REPEAT STEPS 4-8 FOR EACH COUNTER GROUP.

ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.

NOTE: THIS TEST WILL BE SKIPPED UNDER RD CONTROL DUE TO TIMING RESTRICTIONS.

ERROR1: COUNTER REACHED TC IN TOO SHORT A TIME.
ERROR2: COUNTER REACHED TC IN TOO LONG A TIME.

DEBUG: 1) CHECK FOR 5 MHZ SIGNAL AT SRC 2 PIN ON INTERVAL TIMER CHIP.

TEST27: HEADR <^X27> ;THIS TEST SKIPPED IF UNDER RD


```

ZZ-ENKBE-2.1 7000 4
74EC 1586 04B4'DA
74EF 1587
74EF 1588 35 00'00'21
70 C1 04B1'C2
74F8 1589
74F8 1590
74F8 1591 02 3E
74FA 1592 0000'32
74FD 1593
74FD 1594 00B3'CD
7500 1595
7500 1596 C5 46 00'00'21
77 05 3E
7508 1597
7508 1598 00B8'CD
750B 1599
750B 1600 00'5B'21
750E 1601 00D1'CD
7511 1602 0124'CD
7514 1603 0139'CD
7517 1604
7517 1605 00'4F'21
751A 1606 00D1'CD
751D 1607
751D 1608 010F'CD
7520 1609
7520 1610 50 3E
7522 1611 017E'CD
7525 1612
7525 1613 0124'CD
7528 1614
7528 1615 014E'CD
752B 1616 00DD'CD
752E 1617
752E 1618 00'51'21
02 0E 00'8D'11
053F'CA 0000'CD
753C 1619
753C 1620
753C 1621 038A'CD
753F 1622
753F 1623
753F 1624
753F 1625 0000'3A
7542 1626 0F
7543 1627 050B'DA
7546 1628
7546 1629 35 00'00'21
70 C1 0508'C2
754F 1630
754F 1631 3C D3
7551 1632
7551 1633
7551 1634
7551 1635
7551 1636
7551 1637
7551 1638
7551 1639
7551 1640

```

```

*
*
*****

```

```

*****
*
*

```

2\$:

```

* *****
*
*

```

```

* *****
*
*

```

```

*****

```

I 3

JC 1\$

ENDDO

MVI A,2
STA CON_ERR_NUM

CALL CLR.CG_PNT ;CLEAR THE COUNTER GROUP POINTER.

DO 5

CALL INC.CG_PNT ;INCREMENT THE COUNTER GROUP POINTER.

LXI H,TEST27_DAT+4 ;LOAD THE INITIAL COUNT INTO EACH
CALL WRITE_LOAD ; COUNTER.
CALL DISARM_CNTR
CALL LOAD_CNTR

LXI H,ZERO_DAT ;LOAD THE COUNTER RELOAD DATA INTO
CALL WRITE_LOAD ; THE COUNTER LOAD REGISTER.

CALL ARM_CNTR ;START THE COUNTER COUNTING.

MVI A,WAIT_CNT2 ;WAIT A SPECIFIED TIME PERIOD.
CALL WAIT_LOOP

CALL DISARM_CNTR ;STOP THE COUNTER.

CALL SAVE_CNTR ;READ THE COUNTER'S CONTENTS.
CALL READ_HOLD

IFNEQ ONE_SHIFT+1,REC_DAT,2 ;IF THE COUNTER HAS NOT REACHED

; IT'S TERMINAL COUNT...

CALL TEST27_ERROR ;...PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...

RRC
JC 2\$;...JUMP TO BEGINNING OF ERROR LOOP.

ENDDO

TAILR

Fiche 1 Frame I3 Sequence 34
;...JUMP TO BEGINNING OF ERROR LOOP.

TEST28

COUNTER OUTPUT TEST


```

7551 1641
7551 1642
7551 1643
7551 1644
7551 1645
7551 1646
7551 1647
7551 1648
7551 1649
7551 1650
7551 1651
7551 1652
7551 1653
7551 1654
7551 1655
7551 1656
7551 1657
7551 1658
7551 1659
7551 1660
7551 1661
7551 1662
7551 1663
7551 1664
7551 1665
7551 1666
7551 1667
7551 1668
7551 1669
7551 1670
7551 1671
7551 1672
7551 1673
7551 1674
7551 1675
7551 1676

```

```

OF 40D9 3A
28 3E 05D2'DA
0000'3A 0000'CD
C9 0565'D2 OF
OF 0000'3A
3D D3 05D2'DA

```

```

756E 1677
756E 1678 00AE'CD
7571 1679
7571 1680 00'5D'21
7574 1681 00BD'CD
7577 1682
7577 1683 3E 3E
7579 1684 008A'32
757C 1685
757C 1686 00B3'CD
757F 1687
757F 1688 C5 46 00'00'21
77 05 3E
7587 1689
7587 1690 00B8'CD
758A 1691
758A 1692 00'5F'21
758D 1693 00C5'CD
7590 1694

```

```

*****
*
*
*
*
*
*

```

THIS TEST LOADS THE COUNTERS WITH AN INITIAL VALUE, STARTS THE COUNTERS, AND AFTER AN APPROPRIATE WAIT, CHECKS THE COUNTER OUTPUTS WHICH SHOULD BE SET.

TEST STEPS:

- 1) PERFORM MASTER RESET.
- 2) SET UP COUNTER MODE REGISTERS TO COUNT UP, COUNT ONCE, AND HAVE A DELAYED TOGGLE OUTPUT.
- 3) LOAD COUNTER WITH ONE.
- 4) START EACH COUNTER.
- 5) READ THE STATUS REGISTER TO SEE THAT THE COUNTER OUTPUTS HAVE BEEN SET.
- 6) PRINT ERROR IF ANY.

ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.

NOTE: THIS TEST WILL BE SKIPPED UNDER RD CONTROL DUE TO TIMING RESTRICTIONS.

ERROR1: STATUS REGISTER CONTENTS DOESN'T INDICATE COUNTER OUTPUT(S) HAS BEEN SET.

DEBUG: SUSPECT INTERVAL TIMER CHIP.

TEST28: HEADR <^X28> ;THIS TEST SKIPPED IF UNDER RD

1\$:

```

CALL MASTER_RESET ;PERFORM A MASTER RESET.
LXI H,TEST28_DAT ;INITIALIZE THE MASTER MODE REG.
CALL WRITE_MASTER
MVI A,<^X3E> ;LOAD ALL OUTPUT BITS SET DATA
STA EXP_DAT ; INTO EXP_DAT.
CALL CLR.CG_PNT ;CLEAR THE COUNTER GROUP POINTER.
DO 5
CALL INC.CG_PNT ;INCREMENT THE COUNTER GROUP POINTER.
LXI H,TEST28_DAT+2 ;INITIALIZE THE COUNTER MODE REG.
CALL WRITE_MODE

```

```

ZZ-ENKBE-2.1 7000 4
7590 1695 016C'CD
7593 1696
7593 1697 00'51'21
7596 1698 00D1'CD
7599 1699
7599 1700 0124'CD
759C 1701
759C 1702 0139'CD
759F 1703
759F 1704 35 00'00'21
70 C1 0587'C2
75A8 1705
75A8 1706 3F 3E
75AA 1707 1D D3
75AC 1708
75AC 1709 50 3E
75AE 1710 017E'CD
75B1 1711
75B1 1712 1D DB
75B3 1713 3E E6
75B5 1714 008D'32
75B8 1715
75B8 1716 00'8A'21
01 0E 00'8D'11
05C9'CA 0000'CD
75C6 1717
75C6 1718
75C6 1719 03AD'CD
75C9 1720
75C9 1721
75C9 1722
75C9 1723 0000'3A
75CC 1724 0F
75CD 1725 057C'DA
75D0 1726
75D0 1727 3C D3
75D2 1728
75D2 1729
75D2 1730
75D2 1731
75D2 1732
75D2 1733
75D2 1734
75D2 1735
75D2 1736
75D2 1737
75D2 1738
75D2 1739
75D2 1740
75D2 1741
75D2 1742
75D2 1743
75D2 1744
75D2 1745
75D2 1746
75D2 1747
75D2 1748
75D2 1749
75D2 1750
75D2 1751

```

```

*
*
*
*
*
*
*
*
*
*****
*
*
*
*
*
*
*****
*
*
*
*
*
*
*****

```

K 3

Fiche 1 Frame K3 Sequence 36

```

CALL CLR_OUTPUT ;CLEAR THE OUTPUT BIT.
LXI H,ONE_SHIFT+1 ;LOAD A ONE INTO THE COUNTER.
CALL WRITE_LOAD
CALL DISARM_CNTR ;DISARM THE PRESENT COUNTER.
CALL LOAD_CNTR ;LOAD THE COUNTER FROM THE LOAD REG.
ENDDO
MVI A,ARM_ALL ;ARM ALL THE COUNTERS.
OUT ITCSR
MVI A,WAIT_CNT2 ;WAIT A SPECIFIED PERIOD.
CALL WAIT_LOOP
IN ITCSR ;READ THE STATUS REGISTER.
ANI <^X3E> ;MASK OUT BITS 0-2,6 AND 7
STA REC_DAT ;STORE IN REC_DAT.
IFNEQ EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.
CALL TEST28_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1$ ;...JUMP TO BEGINNING OF ERROR LOOP.
TAILR

```

TEST29

COUNTER RELOAD TEST

THIS TEST LOADS THE COUNTERS WITH AN INITIAL VALUE, LOADS THE LOAD REIGISTERS WITH A STRING OF ALTERNATING ONES AND ZEROS, STARTS THE COUNTERS, AND AFTER AN APPROPRIATE WAIT, CHECKS THE COUNTER CONTENTS WHICH SHOULD CONTAIN THE ALTERNATING PATTERN OF ONE AND ZEROS. COUNTER 1 IS ALSO TESTED IN THE GATE CONTROL MODE TO CHECK THAT IT CAN BE RELOADED BY AN ACTIVE EDGE AT GATE 1 OF THE INTERVAL TIMER.

TEST STEPS:

COUNTERS 1 THROUGH 5

1) PERFORM MASTER RESET.

22-ENKBE-2.1 7000 4

75D2 1752
75D2 1753
75D2 1754
75D2 1755
75D2 1756
75D2 1757
75D2 1758
75D2 1759
75D2 1760
75D2 1761
75D2 1762
75D2 1763
75D2 1764
75D2 1765
75D2 1766
75D2 1767
75D2 1768
75D2 1769
75D2 1770
75D2 1771
75D2 1772
75D2 1773
75D2 1774
75D2 1775
75D2 1776
75D2 1777
75D2 1778
75D2 1779
75D2 1780
75D2 1781
75D2 1782
75D2 1783
75D2 1784
75D2 1785
75D2 1786
75D2 1787
75D2 1788
75D2 1789
75D2 1790
75D2 1791
75D2 1792
75D2 1793
75D2 1794
75D2 1795
75D2 1796
75D2 1797
75D2 1798
75D2 1799
75D2 1800
75D2 1801
75D2 1802
75D2 1803
75D2 1804
75D2 1805
75D2 1806
75D2 1807
75D2 1808
75D2 1809
75D2 1810
75D2 1811

L 3

Fiche 1 Frame L3

Sequence 37

- 2) SET UP COUNTER MODE REGISTERS TO COUNT UP, COUNT ONCE, AND RELOAD FROM THE LOAD REGISTER.
- 3) LOAD COUNTER WITH ONE.
- 4) LOAD LOAD REGISTERS WITH HEX 5555 (ALTERNATING ONES AND ZEROS).
- 5) START EACH COUNTER.
- 7) SAVE EACH COUNTER IN ITS HOLD REGISTER.
- 8) READ EACH HOLD REGISTER AND COMPARE WITH HEX 5555.
- 9) PRINT ERROR IF ANY.

COUNTER 1 (RELOAD FROM GATE EDGE)

- 1) PERFORM A MASTER RESET.
- 2) SET UP MASTER MODE REGISTER TO COMPARE ENABLE, ETC..
- 3) SET UP COUNTERS 1 AND 2 TO COUNT UP, COUNT REPETITIVELY, RELOAD FROM THE LOAD REGISTER, AND START COUNTING ON ACTIVE GATE LEVEL.
- 4) LOAD COUNTER 1 WITH ZERO AND LOAD ITS ALARM REGISTER WITH ONE.
- 5) WRITE THE RELOAD DATA.
- 6) LOAD COUNTER 2 WITH ZERO, LOAD ITS ALARM WITH ZERO, AND ARM THE COUNTER.
- 7) ENABLE COUNTER 1 WITH ACTIVE GATE LEVEL AND START IT COUNTING BY ARMING IT.
- 8) WAIT FOR GATE EDGE AND THEN CHECK THAT COUNTER 1 HAS BEEN RELOADED.
- 9) PRINT ERROR IF ANY.

ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.

NOTE: THIS TEST WILL BE SKIPPED UNDER RD CONTROL DUE TO TIMING RESTRICTIONS.

ERROR1: COUNTER HAS NOT BEEN RELOADED CORRECTLY. ADDITIONAL DATA WILL INDICATE WHICH COUNTER GROUP.

ERROR2: COUNTER 1 HAS NOT BEEN RELOADED CORRECTLY IN GATE CONTROL MODE. ADDITIONAL DATA WILL INDICATE COUNTER GROUP 1.

DEBUG:

FOR COUNTERS 1 THROUGH 5

- 1) SUSPECT INTERVAL TIMER CHIP.

FOR COUNTER 1 (RELOAD FROM GATE EDGE)

- 1) CHECK THE FOLLOWING PINS ON CHIP E32 AFTER THE SECOND SIM INSTRUCTION, BUT BEFORE THE OUT SETIMR INSTRUCTION; PIN 12 SHOULD BE HIGH AND PIN 13 LOW, PIN 2 SHOULD BE HIGH AND PIN 1 LOW. PIN 5 OF CHIP E17 SHOULD BE HIGH AT THIS SAME TIME. IF ANY OF THESE VALUES ARE INCORRECT CHECK THAT THE VALUES PROPAGATE CORRECTLY THROUGH CHIPS E32, E41, E42, E144.
- 2) AFTER THE CALL ARM CNTR INSTRUCTION FOR COUNTER CHECK THE FOLLOWING VALUES. PIN 13 OF CHIP E32 SHOULD GO HIGH CAUSING PIN 11 TO GO LOW. THIS SHOULD CAUSE 'WCSB TIMER INT L' TO GO LOW AND PIN 6 OF CHIP E17 TO PULSE LOW. IF ANYTHING IS

ZZ-ENKBE-2.1 7000 4

75D2 1812
75D2 1813
75D2 1814
75D2 1815
75D2 1816
75D2 1817
75D2 1818
75D2 1819
75D2 1820
75D2 1821
75D2 1822
75D2 1823

OF 40D9 3A
29 3E 070B'DA
0000'3A 0000'CD
C9 05E6'D2 OF
OF 0000'3A
3D D3 070B'DA

75EF 1824
75EF 1825 00AE'CD
75F2 1826
75F2 1827 00'61'21
75F5 1828 00BD'CD
75F8 1829
75F8 1830 00B3'CD
75FB 1831
75FB 1832 C5 46 00'00'21
77 05 3E

7603 1833
7603 1834 00B8'CD
7606 1835
7606 1836 00'63'21
7609 1837 00C5'CD
760C 1838
760C 1839 35 00'00'21
70 C1 0603'C2

7615 1840
7615 1841 01 3E
7617 1842 0000'32
761A 1843
761A 1844 0065'2A
761D 1845 008A'22
7620 1846
7620 1847 00B3'CD
7623 1848
7623 1849 C5 46 00'00'21
77 05 3E

762B 1850
762B 1851 00B8'CD
762E 1852
762E 1853 00'51'21
7631 1854 00D1'CD
7634 1855
7634 1856 0124'CD
7637 1857
7637 1858 0139'CD
763A 1859
763A 1860 0065'2A
763D 1861 000E'22
7640 1862 00'0E'21
7643 1863 0000'CD

M 3

Fiche 1 Frame M3

Sequence 38

WRONG AT THIS POINT CHECK CHIPS E32, E37, E41, E42, E144.

3) CHECK CHIPS E3, E41, E42, E43 AT THE APPROPRIATE PINS FOR THE PRESENCE OF 'WCSB 1 MHZ H', A 1 μ S CLOCK.

4) CHECK PIN 1 OF CHIP E 43 FOR 'WCSA CLK L', A 5 MHZ SIGNAL.

TEST29: HEADR <^X29> ;THIS TEST SKIPPED IF UNDER RD

CALL MASTER_RESET ;PERFORM A MASTER RESET.
LXI H,TEST29_DAT ;INITIALIZE THE MASTER MODE REG.
CALL WRITE_MASTER
CALL CLR_CG_PNT ;CLEAR THE COUNTER GROUP POINTER.
DO 5

CALL INC_CG_PNT ;INCREMENT THE COUNTER GROUP POINTER.
LXI H,TEST29_DAT+2 ;INITIALIZE THE COUNTER MODE REG.
CALL WRITE_MODE
ENDDO

MVI A,1 ;FIRST PART OF TEST 29.
STA CON_ERR_NUM
LHLD TEST29_DAT+4 ;LOAD ALTERNATING PATTERN OF ONES AND
SHLD EXP_DAT ;ZEROS INTO EXP_DAT.
CALL CLR_CG_PNT ;CLEAR THE COUNTER GROUP POINTER.
DO 5

CALL INC_CG_PNT ;INCREMENT THE COUNTER GROUP POINTER.
LXI H,ONE_SHIFT+1 ;LOAD A ONE INTO THE LOAD REGISTER.
CALL WRITE_LOAD
CALL DISARM_CNTR ;DISARM THE PRESENT COUNTER.
CALL LOAD_CNTR ;LOAD THE COUNTER FROM THE LOAD REG.
LHLD TEST29_DAT+4 ;LOAD ALTERNATING PATTERN OF ONES AND
SHLD TIMER_DAT ;ZEROS INTO TIMER DAT. DECREMENT THIS
LXI H,TIMER_DAT ;PATTERN AND WRITE IT TO THE LOAD REG.
CALL DECR_WORD

1\$:

ZZ-ENKBE-2.1	7000	4
7646 1864	00'0E'21	
7649 1865	00D1'CD	
764C 1866		
764C 1867	010F'CD	
764F 1868		
764F 1869	50 3E	
7651 1870	017E'CD	
7654 1871		
7654 1872	014E'CD	
7657 1873		
7657 1874	00DD'CD	
765A 1875		
765A 1876	00'8A'21	
	02 0E 00'8D'11	
	066B'CA 0000'CD	
7668 1877		
7668 1878		
7668 1879	03D2'CD	
766B 1880		
766B 1881		
766B 1882		
766B 1883	0000'3A	
766E 1884	0F	
766F 1885	062E'DA	
7672 1886		
7672 1887	35 00'00'21	
	70 C1 062B'C2	
767B 1888		
767B 1889		
767B 1890		
767B 1891	02 3E	
767D 1892	0000'32	
7680 1893		
7680 1894	0052'2A	
7683 1895	008A'22	
7686 1896		
7686 1897	00AE'CD	
7689 1898		
7689 1899	00'67'21	
768C 1900	00BD'CD	
768F 1901		
768F 1902	01 3E	
7691 1903	000C'32	
7694 1904		
7694 1905	00'69'21	
7697 1906	00C5'CD	
769A 1907		
769A 1908	016C'CD	
769D 1909		
769D 1910	00'4F'21	
76A0 1911	00D1'CD	
76A3 1912		
76A3 1913	00'51'21	
76A6 1914	00E9'CD	
76A9 1915		
76A9 1916	0139'CD	
76AC 1917		
76AC 1918	00'52'21	
76AF 1919	00D1'CD	
76B2 1920		

N 3

Fiche 1 Frame N3

Sequence 39

```

LXI      H,TIMER_DAT      ;TIMER DAT
CALL     WRITE_LOAD        ;WRITE TO LOAD REGISTER.

CALL     ARM_CNTR          ;ARM THE COUNTER.

MVI      A,WAIT_CNT2       ;ENTER A WAIT LOOP.
CALL     WAIT_LOOP

CALL     SAVE_CNTR         ;SAVE THE COUNTER IN THE HOLD REG.

CALL     READ_HOLD         ;READ THE HOLD REGISTER.

IFNEQ    EXP_DAT,REC_DAT,2  ;COMPARE THE EXPECTED AND
                             ; RECEIVED DATA.

CALL     TEST29_ERROR      ;PRINT AN ERROR MESSAGE.

ENDIF

LDA      ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC       1$                ;...JUMP TO BEGINNING OF ERROR LOOP.

ENDDO

MVI      A,2               ;SECOND PART OF TEST 29.
STA      CON_ERR_NUM

LHLD     ONE_SHIFT+2       ;LOAD THE COUNTER RELOAD DATA
SHLD     EXP_DAT           ; INTO EXP_DAT.

CALL     MASTER_RESET      ;PERFORM A MASTER RESET.

LXI      H,TEST29_DAT+6    ;WRITE THE MASTER MODE DATA TO THE
CALL     WRITE_MASTER      ; MASTER MODE REGISTER.

MVI      A,1               ;POINT TO COUNTER GROUP 1.
STA      CG_PNT

LXI      H,TEST29_DAT+8    ;WRITE THE MODE DATA.
CALL     WRITE_MODE

CALL     CLR_OUTPUT        ;CLEAR THE OUTPUT BIT.

LXI      H,ZERO_DAT        ;WRITE THE COUNTER INITIAL VALUE.
CALL     WRITE_LOAD

LXI      H,ONE_SHIFT+1     ;WRITE THE ALARM DATA.
CALL     WRITE_ALARM

CALL     LOAD_CNTR         ;LOAD COUNTER 1.

LXI      H,ONE_SHIFT+2     ;WRITE THE COUNTER RELOAD DATA
CALL     WRITE_LOAD        ; TO THE LOAD REGISTER.

```

```

ZZ-ENKBE-2.1 7000 4
76B2 1921 00B8'CD
76B5 1922
76B5 1923 00'6B'21
76B8 1924 00C5'CD
76BB 1925
76BB 1926 016C'CD
76BE 1927
76BE 1928 00'4F'21
76C1 1929 00D1'CD
76C4 1930
76C4 1931 00'4F'21
76C7 1932 00E9'CD
76CA 1933
76CA 1934 0139'CD
76CD 1935 010F'CD
76D0 1936
76D0 1937 00B3'CD
76D3 1938 00B8'CD
76D6 1939
76D6 1940 40 3E
76D8 1941 30
76D9 1942 C0 3E
76DB 1943 30
76DC 1944
76DC 1945 00
76DD 1946 00
76DE 1947
76DE 1948 2D D3
76E0 1949
76E0 1950 010F'CD
76E3 1951
76E3 1952 00
76E4 1953 00
76E5 1954 00
76E6 1955
76E6 1956 0124'CD
76E9 1957
76E9 1958 014E'CD
76EC 1959 00DD'CD
76EF 1960
76EF 1961 2C D3
76F1 1962
76F1 1963 00'8A'21
02 0E 00'8D'11
0702'F2 0000'CD

76FF 1964
76FF 1965
76FF 1966 03D2'CD
7702 1967
7702 1968
7702 1969
7702 1970 0000'3A
7705 1971 0F
7706 1972 0686'DA
7709 1973
7709 1974 3C D3
770B 1975
770B 1976
770B 1977
770B 1978

```

```

*****
*
*
*
*
*
*****

```

```

B 4
Fiche 1 Frame B4 Sequence 40
CALL INC_CG_PNT ;POINT TO COUNTER GROUP 2.
LXI H,TEST29 DAT+10 ;WRITE THE MODE DATA.
CALL WRITE_MODE
CALL CLR_OUTPUT ;CLEAR THE OUTPUT BIT.
LXI H,ZERO_DAT ;WRITE THE COUNTER INITIAL VALUE.
CALL WRITE_LOAD
LXI H,ZERO_DAT ;WRITE THE ALARM DATA.
CALL WRITE_ALARM
CALL LOAD_CNTR ;LOAD AND ARM COUNTER 2.
CALL ARM_CNTR
CALL CLR_CG_PNT ;POINT TO COUNTER GROUP 1.
CALL INC_CG_PNT
MVI A,CLR_SER_OUT ;CLEAR THE 8085 SERIAL OUTPUT.
SIM
MVI A,SET_SER_OUT ;SET THE 8085 SERIAL OUTPUT TO
SIM ; ENABLE A TIMER INTERRUPT.
NOP ;ALLOW TIME FOR TIMER CONTROL
NOP ; SIGNALS TO SETTLE.
OUT SETTMR ;ENABLE TIMER COUNTING.
CALL ARM_CNTR ;ARM COUNTER 1.
NOP ;WAIT FOR ALARM REGISTERS TO GO OFF
NOP ; AND ACTIVE EDGE AT GATE 1 TO
NOP ; RELOAD COUNTER 1.
CALL DISARM_CNTR ;DISARM COUNTER 1.
CALL SAVE_CNTR ;READ THE CONTENTS OF COUNTER 1.
CALL READ_HOLD
OUT CLRTMR ;STOP TIMER.
IFGT EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.
CALL TEST29_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 2$ ;...JUMP TO BEGINNING OF ERROR LOOP.
TAILR

```

```

:*****
:
:

```


770B 1979
770B 1980
770B 1981
770B 1982
770B 1983
770B 1984
770B 1985
770B 1986
770B 1987
770B 1988
770B 1989
770B 1990
770B 1991
770B 1992
770B 1993
770B 1994
770B 1995
770B 1996
770B 1997
770B 1998
770B 1999
770B 2000
770B 2001
770B 2002
770B 2003
770B 2004
770B 2005
770B 2006
770B 2007
770B 2008
770B 2009
770B 2010
770B 2011
770B 2012
770B 2013
770B 2014
770B 2015
770B 2016
770B 2017
770B 2018
770B 2019
770B 2020
770B 2021
770B 2022
770B 2023
770B 2024
770B 2025
770B 2026
770B 2027
770B 2028
770B 2029
770B 2030
770B 2031
770B 2032
770B 2033
770B 2034
770B 2035
770B 2036
770B 2037
770B 2038

TEST2A

COUNTER CONCATENATION TEST

THIS TEST DETERMINES WHETHER COUNTERS 1 AND 2 AND COUNTERS 4 AND 5 CAN BE PROPERLY CONCATENATED. COUNTERS 1 AND 2 WILL BE LOADED WITH ONES AND COUNTER 2 WILL BE SET UP TO COUNT ON THE TC PULSE OF COUNTER 1. COUNTERS 1 AND 2 WILL BE STARTED AND AFTER A WAIT LOOP TO ALLOW COUNTER 1 TO REACH TC ONCE, THE COUNTER 2 CONTENTS WILL BE CHECKED FOR A VALUE OF ZERO.

THE SAME THING WILL BE DONE FOR COUNTER PAIR 4 AND 5 EXCEPT THAT COUNTER 4 WILL BE INITIALIZED WITH A HIGH COUNT VALUE AND WILL COUNT FROM THE 100 HZ CLOCK AT THE SRC 1 PIN. BEFORE THE TEST FOR COUNTER PAIR 4 AND 5 THE PRESENCE OF THE 100 HZ CLOCK WILL BE CHECKED.

TEST STEPS:

- 1) PERFORM MASTER RESET
- 2) SET UP COUNTER MODE REGISTERS 1 AND 4 TO COUNT UP, COUNT REPETITIVELY, AND HAVE A TC PULSE OUTPUT.
- 3) SET UP COUNTER MODE REGISTERS 2 AND 5 TO COUNT UP ON THE TC PULSE OF THE NEXT LOWER COUNTER.
- 4) LOAD ONES INTO COUNTERS 1 AND 2.
- 6) START COUNTERS 1 AND 2.
- 7) AFTER AN APPROPRIATE WAIT LOOP CHECK COUNTER 2 FOR A TWO.
- 8) PRINT ERRORS IF ANY.
- 9) CHECK 100 HZ SIGNAL BY SELECTING IT ON CHIP E29 AND WATCH FOR A CHANGE ON 'BUS AD7 H'.
- 10) PRINT ERROR IF ANY.
- 11) REPEAT STEPS 4-8 FOR COUNTER PAIR 4/5 WITH CHANGES MENTIONED ABOVE.

ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.

NOTE: THIS TEST WILL BE SKIPPED UNDER RD CONTROL DUE TO TIMING RESTRICTIONS.

ERROR1: COUNTER 2 HAS NOT BEEN INCREMENTED BY COUNTER 1.
ERROR2: 100 HZ CLOCK CANNOT BE READ ON 'BUS AD7 H'.
ERROR3: COUNTER 5 HAS NOT BEEN INCREMENTED BY COUNTER 4.

DEBUG:

FOR COUNTER CONCATENATION TESTS

- 1) SUSPECT INTERVAL TIMER CHIP.

FOR 100 HZ CLOCK TEST

- 1) CHECK PIN 7 OF THE INTERVAL TIMER CHIP FOR A 100 HZ SIGNAL.
- 2) CHECK PIN 33 OF THE INTERVAL TIMER CHIP FOR THE SAME 100 HZ SIGNAL.

ZZ-ENKBE-2.1 7000 4
770B 2039
770B 2040
770B 2041
770B 2042
770B 2043
770B 2044
770B 2045
770B 2046
770B 2047
770B 2048

OF 40D9 3A
2A 3E 0836'DA
0000'3A 0000'CD
C9 071F'D2 OF
OF 0000'3A
3D D3 0836'DA

7728 2049
7728 2050 00AE'CD
772B 2051
772B 2052 00'6D'21
772E 2053 00BD'CD
7731 2054
7731 2055 01 3E
7733 2056 000C'32
7736 2057 00'6F'21
7739 2058 00C5'CD
773C 2059
773C 2060 00B8'CD
773F 2061
773F 2062 00'71'21
7742 2063 00C5'CD
7745 2064
7745 2065 00B8'CD
7748 2066 00B8'CD
774B 2067
774B 2068 00'73'21
774E 2069 00C5'CD
7751 2070
7751 2071 00B8'CD
7754 2072
7754 2073 00'75'21
7757 2074 00C5'CD
775A 2075
775A 2076 0079'2A
775D 2077 008A'22
7760 2078
7760 2079 01 3E
7762 2080 0000'32
7765 2081
7765 2082 02 3E
7767 2083 000C'32
776A 2084 00'51'21
776D 2085 00D1'CD
7770 2086
7770 2087 0124'CD
7773 2088 0139'CD
7776 2089 010F'CD
7779 2090
7779 2091 01 3E
777B 2092 000C'32
777E 2093 00'51'21

D 4

Fiche 1 Frame D4

Sequence 42

- 3) CHECK 'WCSA A10 H' THROUGH 'WCSA A08 H' FOR HEX 6 ON
CHIP E29 SELECTING 'WCSB LAT 100HZ H' ON 'BUS AD7 H'.
4) CHECK PIN 13 OF CHIP E99 FOR A HIGH.

TEST2A:

HEADR <^X2A>

;THIS TEST SKIPPED IF UNDER RD

CALL MASTER_RESET ;PERFORM A MASTER RESET.
LXI H,TEST2A_DAT ;INITIALIZE THE MASTER MODE REG.
CALL WRITE_MASTER
MVI A,1
STA CG_PNT
LXI H,TEST2A_DAT+2 ;INITIALIZE THE COUNTER 1 MODE REG.
CALL WRITE_MODE
CALL INC_CG_PNT
LXI H,TEST2A_DAT+4 ;INITIALIZE THE COUNTER 2 MODE REG.
CALL WRITE_MODE
CALL INC_CG_PNT
CALL INC_CG_PNT
LXI H,TEST2A_DAT+6 ;INITIALIZE THE COUNTER 4 MODE REG.
CALL WRITE_MODE
CALL INC_CG_PNT
LXI H,TEST2A_DAT+8 ;INITIALIZE THE COUNTER 5 MODE REG.
CALL WRITE_MODE
LHLD TEST2A_DAT+12 ;LOAD A TWO INTO EXP_DAT.
SHLD EXP_DAT
MVI A,1 ;FIRST PART OF TEST 2A.
STA CON_ERR_NUM
MVI A,2 ;LOAD A ONE INTO THE COUNTER 2
STA CG_PNT ;LOAD REGISTER.
LXI H,ONE_SHIFT+1
CALL WRITE_LOAD
CALL DISARM_CNTR ;LOAD AND ARM COUNTER 2.
CALL LOAD_CNTR
CALL ARM_CNTR
MVI A,1 ;LOAD HEX FFFF INTO COUNTER 1.
STA CG_PNT
LXI H,ONE_SHIFT+1

1\$:


```

ZZ-ENKBE-2.1 7000 4
7781 2094 00D1'CD
7784 2095 0139'CD
7787 2096 010F'CD
778A 2097
778A 2098 50 3E
778C 2099 017E'CD
778F 2100
778F 2101 02 3E
7791 2102 000C'32
7794 2103 014E'CD
7797 2104 00DD'CD
779A 2105
779A 2106 00'8A'21
              02 0E 00'8D'11
              07AB'CA 0000'CD

77A8 2107
77A8 2108
77A8 2109 03F3'CD
77AB 2110
77AB 2111
77AB 2112
77AB 2113 0000'3A
77AE 2114 0F
77AF 2115 0765'DA
77B2 2116
77B2 2117 02 3E
77B4 2118 0000'32
77B7 2119
77B7 2120 06 DB
77B9 2121 80 E6
77BB 2122
77BB 2123 4F
77BC 2124
77BC 2125 10 3E
77BE 2126 0011'32
77C1 2127
77C1 2128 01 3E
77C3 2129 017E'CD
77C6 2130
77C6 2131 06 DB
77C8 2132 80 E6
77CA 2133
77CA 2134 B9
77CB 2135 07E2'C2
77CE 2136
77CE 2137 0011'3A
77D1 2138 3D
77D2 2139 0011'32
77D5 2140
77D5 2141 07C1'C2
77D8 2142
77D8 2143
77D8 2144 0407'CD
77DB 2145
77DB 2146 0000'3A
77DE 2147 0F
77DF 2148 07B7'DA
77E2 2149
77E2 2150 03 3E
77E4 2151 0000'32

```

```

*****
*
*
*
*
*
*****

```

2\$:

3\$:

4\$:

E 4

Fiche 1 Frame E4

Sequence 43

```

CALL WRITE_LOAD
CALL LOAD_CNTR
CALL ARM_CNTR

MVI A, WAIT_CNT2 ;WAIT FOR COUNTER 1 TO REACH TC.
CALL WAIT_LOOP

MVI A, 2 ;SAVE THE CONTENTS OF COUNTER 2 IN
STA CG_PNT ; REC_DAT.
CALL SAVE_CNTR
CALL READ_HOLD

IFNEQ EXP_DAT, REC_DAT, 2 ;COMPARE THE EXPECTED AND
                                ; RECEIVED DATA.

CALL TEST2A_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1$ ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI A, 2 ;SECOND PART OF TEST 2A.
STA CON_ERR_NUM

IN LAT100HZ ;READ 100 HZ CLOCK.
ANI <^X80>

MOV C, A ;REMEMBER CLOCK STATE IN REGISTER C.

MVI A, <^X10> ;LOOP CHECKING FOR CLOCK TO CHANGE.
STA COUNT_SAVE

MVI A, 1 ;INCREASE LOOP TIME.
CALL WAIT_LOOP

IN LAT100HZ ;READ 100 HZ CLOCK.
ANI <^X80>

CMP C ;IF CHANGE IS DETECTED BEFORE COUNT IS
JNZ 4$ ; UP THEN JUMP TO THIRD PART OF 2A.

LDA COUNT_SAVE ;DECREMENT LOOP COUNT.
DCR A
STA COUNT_SAVE

JNZ 3$ ;IF COUNT IS UP AND A CLOCK EDGE WAS
; NOT DETECTED...

CALL TEST2A_2_ERROR ;...PRINT ERROR.

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 2$ ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI A, 3 ;THIRD PART OF TEST 2A.
STA CON_ERR_NUM

```

ZZ-ENKBE-2.1 7000 4

F 4

Fiche 1 Frame F4

Sequence 44

77E7 2152
77E7 2153 05 3E
77E9 2154 000C'32
77EC 2155 00'51'21
77EF 2156 00D1'CD
77F2 2157
77F2 2158 0124'CD
77F5 2159 0139'CD
77F8 2160 010F'CD
77FB 2161
77FB 2162 04 3E
77FD 2163 000C'32
7800 2164 00'77'21
7803 2165 00D1'CD
7806 2166 0139'CD
7809 2167 010F'CD
780C 2168
780C 2169 C0 3E
780E 2170 017E'CD
7811 2171
7811 2172 05 3E
7813 2173 000C'32
7816 2174 014E'CD
7819 2175 00DD'CD
781C 2176
781C 2177 00'8A'21
02 0E 00'8D'11
082D'CA 0000'CD
782A 2178
782A 2179
782A 2180 03F3'CD
782D 2181
782D 2182
782D 2183
782D 2184 0000'3A
7830 2185 0F
7831 2186 07E7'DA
7834 2187
7834 2188 3C D3
7836 2189
7836 2190
7836 2191
7836 2192
7836 2193
7836 2194
7836 2195
7836 2196
7836 2197
7836 2198
7836 2199
7836 2200
7836 2201
7836 2202
7836 2203
7836 2204
7836 2205
7836 2206
7836 2207
7836 2208
7836 2209

*
*
*
*
*

5\$:

```

MVI A,5 ;LOAD A ONE INTO THE COUNTER 5
STA CG_PNT ; LOAD REGISTER.
LXI H,ONE_SHIFT+1
CALL WRITE_LOAD

CALL DISARM_CNTR ;LOAD AND ARM COUNTER 5.
CALL LOAD_CNTR
CALL ARM_CNTR

MVI A,4 ;LOAD HEX FFFF INTO COUNTER 4.
STA CG_PNT
LXI H,TEST2A_DAT+10
CALL WRITE_LOAD
CALL LOAD_CNTR
CALL ARM_CNTR

MVI A,WAIT_CNT3 ;WAIT FOR COUNTER 4 TO REACH TC.
CALL WAIT_LOOP

MVI A,5 ;SAVE THE CONTENTS OF COUNTER 5 IN
STA CG_PNT ; REC_DAT.
CALL SAVE_CNTR
CALL READ_HOLD ;READ THE HOLD REGISTER.

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.

CALL TEST2A_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 5$ ;...JUMP TO BEGINNING OF ERROR LOOP.

TAILR

```

TEST2B

SUMMARY REGISTER TEST (TIMER INPUTS)

BITS 6 AND 7 OF THE SUMMARY REGISTER ARE TESTED BY ASSERTING BOTH THE TIMER INTERRUPT AND POWER FAIL SIGNALS FROM THE INTERVAL TIMER AND CHECKING TO SEE THAT THE CORRESPONDING BITS OF THE SUMMARY REGISTER ARE ASSERTED. THESE SIGNALS ARE THEN REMOVED AND THE SUMMARY REGISTER IS AGAIN CHECKED. IN ADDITION TO READING THE SUMMARY REGISTER WHEN THE TIMER INTERRUPT IS ASSERTED THE 8085 RST 5.5 STATUS IS READ TO CHECK THAT THE 8085 CAN BE INTERRUPTED BY THE TIMER. THIS TEST IS DONE WITH RST 5.5 DISABLED.

TEST STEPS:

TIMER INTERRUPT BIT

7836 2210
7836 2211
7836 2212
7836 2213
7836 2214
7836 2215
7836 2216
7836 2217
7836 2218
7836 2219
7836 2220
7836 2221
7836 2222
7836 2223
7836 2224
7836 2225
7836 2226
7836 2227
7836 2228
7836 2229
7836 2230
7836 2231
7836 2232
7836 2233
7836 2234
7836 2235
7836 2236
7836 2237
7836 2238
7836 2239
7836 2240
7836 2241
7836 2242
7836 2243
7836 2244
7836 2245
7836 2246
7836 2247
7836 2248
7836 2249
7836 2250
7836 2251
7836 2252
7836 2253
7836 2254
7836 2255
7836 2256
7836 2257
7836 2258
7836 2259
7836 2260
7836 2261
7836 2262
7836 2263
7836 2264
7836 2265
7836 2266
7836 2267
7836 2268
7836 2269

- 1) PERFORM A MASTER RESET.
- 2) SET UP MASTER MODE REGISTER TO COMPARE ENABLE, ETC.
- 3) SET UP COUNTERS 1 AND 2 TO COUNT UP, REPETITIVELY, ETC.
- 4) LOAD ZEROS INTO LOAD REGISTERS 1 AND 2.
- 5) WRITE A ONE TO ALARM 1 AND A ZERO TO ALARM 2.
- 6) LOAD AND ARM THE COUNTERS.
- 7) CLEAR THEN SET THE 8085 SERIAL OUTPUT TO ENABLE A TIMER INTERRUPT.
- 8) READ 8085 RST 5.5 STATUS.
- 9) PRINT ERROR IF NOT CLEARED.
- 10) TURN ON COUNTER 1 BY ASSERTING ITS INPUT GATE.
- 11) WAIT FOR A SPECIFIED PERIOD OF TIME TO ALLOW TIMER INTERRUPT TO BE ASSERTED (LOW).
- 12) READ THE SUMMARY REGISTER AND CHECK FOR BIT 7 LOW.
- 13) PRINT ERROR IF ANY.
- 14) READ 8085 RST 5.5 STATUS.
- 15) PRINT ERROR IF NOT ASSERTED.
- 16) CLEAR THE 8085 SERIAL OUTPUT.
- 17) READ THE SUMMARY REGISTER AND CHECK FOR BIT 7 HIGH.
- 18) PRINT ERROR IF ANY.

POWER FAIL BIT

- 1) SET MODE REGISTER 3 TO COUNT UP, ONCE, NO GATING, ETC.
- 2) LOAD A ONE INTO COUNTER 3.
- 3) CLEAR THE COUNTER'S OUTPUT BIT.
- 4) ARM THE COUNTER.
- 5) WAIT A SPECIFIED TIMER PERIOD FOR THE POWER FAIL BIT TO BE ASSERTED.
- 6) READ THE SUMMARY REGISTER AND CHECK FOR BIT 6 LOW.
- 7) PRINT ERROR IF ANY.
- 8) CLEAR THE OUTPUT BIT.
- 9) READ THE SUMMARY REGISTER AND CHECK FOR BIT 6 HIGH.
- 10) PRINT ERROR IF ANY.

ASSUMPTIONS: 1) PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.
2) SINCE THE TEST IS RUN WITH INTERRUPTS DISABLED, IT IS ASSUMED THAT RST 5.5 WILL BE SERVICED SHOULD AN INTERRUPT REQUEST OCCUR.

NOTE: THIS TEST WILL BE SKIPPED UNDER RD CONTROL DUE TO TIMING RESTRICTIONS.

ERROR1: RST 5.5 NOT CLEARED.
ERROR2: SUMMARY REGISTER BIT 7 (TIMER INT) NOT ASSERTED (LOW).
ERROR3: RST 5.5 NOT ASSERTED.
ERROR4: SUMMARY REGISTER BIT 7 (TIMER INT) NOT CLEARED (HIGH).
ERROR5: SUMMARY REGISTER BIT 6 (POWER FAIL) NOT ASSERTED (LOW).
ERROR6: SUMMARY REGISTER BIT 6 (POWER FAIL) NOT CLEARED (HIGH).

DEBUG:

FOR TIMER INTERRUPT BIT

- 1) CHECK 'WCSA 8085 SER OUT H' FOR A LOW VALUE AFTER FIRST SIM INSTRUCTION IN FIRST PART TEST B.
- 2) CHECK PIN 5 OF 7400 CHIP E17 FOR A HIGH VALUE IN FIRST PART OF TEST 2B AFTER THE SECOND SIM BUT BEFORE THE OUT

```

ZZ-ENKBE-2.1  7000  4
7836 2270
7836 2271
7836 2272
7836 2273
7836 2274
7836 2275
7836 2276
7836 2277
7836 2278
7836 2279
7836 2280
7836 2281
7836 2282
7836 2283
7836 2284
7836 2285
7836 2286
7836 2287
7836 2288
7836 2289
7836 2290
7836 2291
7836 2292
7836 2293
7836 2294
7836 2295
7836 2296
7836 2297
7836 2298
7836 2299
7836 2300
7836 2301
7836 2302
7836 2303
7836 2304
7836 2305
7836 2306
7836 2307 OF 40D9 3A
              2B 3E 0975'DA
              0000'3A 0000'CD
              C9 084A'D2 OF
              OF 0000'3A
              3D D3 0975'DA
7853 2308
7853 2309 01 3E
7855 2310 0000'32
7858 2311
7858 2312 00 3E
785A 2313 008A'32
785D 2314
785D 2315 00AE'CD
7860 2316
7860 2317 00'7B'21
7863 2318 00BD'CD
7866 2319
7866 2320 01 3E
7868 2321 000C'32
786B 2322
786B 2323 00'7D'21
786E 2324 00C5'CD

```

H 4

Fiche 1 Frame H4

Sequence 46

SETTMR INSTRUCTION. AT THIS SAME TIME PINS 12 AND 13 OF CHIP E32 SHOULD BE HIGH AND LOW RESPECTIVELY, 'WCSA 8085 SER OUT H' SHOULD BE HIGH, AND PINS 1 AND 2 OF THE 7400 CHIP E32 SHOULD BE LOW AND HIGH RESPECTIVELY. IF A PROBLEM OCCURS HERE, CHECK CHIPS E41, E42, E144 AND E32.

- 3) CHECK 'WCSB START INTRVL TMR H' FOR A HIGH VALUE AFTER THE OUT SETTMR INSTRUCTION BUT BEFORE THE OUT CLRTMR INSTRUCTION. IF AN ERROR OCCURS HERE CHECK CHIP E37.
- 4) AFTER THE OUT SETTMR INSTRUCTION, PIN 13 OF CHIP E32 SHOULD GO HIGH, CAUSING PIN 11 TO GO LOW. THIS SHOULD CAUSE THE SIGNALS 'WCSB TIMER INT L' TO GO LOW, 'WCSB TIMER INTR H' TO GO HIGH, AND PIN 5 OF CHIP E17 TO GO HIGH.
- 5) CHECK CHIP E23 PIN 13 FOR 'WCSB TIMER INT L' TO GO LOW, AND CHIP E4 PIN 9 TO GO HIGH AFTER THE SETTMR INSTRUCTION.
- 6) CHECK 'WCSA SEL ROM MUX L' FOR LOW VALUE DURING IN SUMREG INSTRUCTION.
- 7) CHECK CHIP E21 (PAL 16L8).
- 8) CHECK 'WCSB START INTERVAL TIMER H' FOR LOW VALUE AFTER THE CLRTMR INSTRUCTION IN THE SECOND PART OF TEST B.

FOR POWER FAIL BIT

- 1) CHECK 'WCSB PWR FAIL TMR L' FOR HIGH VALUE AFTER CLR_OUTPUT ROUTINE IN THIRD PART OF TEST 2B.
- 2) CHECK 'WCSB PWR FAIL TMR L' FOR LOW VALUE AFTER WAIT LOOP IN THIRD PART OF TEST 2B.
- 3) CHECK CHIP E23 PIN 10 FOR 'WCSB PWR FAIL TMR INT L' TO GO LOW AFTER THE WAIT LOOP.
- 4) CHECK CHIP E23 PIN 10 FOR 'WCSB PWR FAIL TMR INT L' TO GO HIGH AFTER CLR_OUTPUT ROUTINE IN TEST 2B PART 4.

TEST2B: HEADR <^X2B> ;THIS TEST SKIPPED IF UNDER RD

```

1$:                MVI        A,1                ;FIRST PART OF TEST 2B.
                   STA        CON_ERR_NUM
                   MVI        A,0                ;LOAD RST 5.5 NOT REQUESTED
                   STA        EXP_DAT            ; DATA INTO EXP_DAT.
2$:                CALL        MASTER_RESET      ;PERFORM A MASTER RESET.
                   LXI        H,TEST2B DAT      ;WRITE THE MASTER MODE DATA TO THE
                   CALL        WRITE_MASTER      ; MASTER MODE REGISTER.
                   MVI        A,1                ;POINT TO COUNTER GROUP 1.
                   STA        CG_PNT
                   LXI        H,TEST2B DAT+2     ;WRITE THE MODE DATA.
                   CALL        WRITE_MODE

```



```

ZZ-ENKBE-2.1  7000  4
7871 2325
7871 2326 016C'CD
7874 2327
7874 2328 00'4F'21
7877 2329 00D1'CD
787A 2330
787A 2331 00'51'21
787D 2332 00E9'CD
7880 2333
7880 2334 0139'CD
7883 2335 010F'CD
7886 2336
7886 2337 00B8'CD
7889 2338
7889 2339 00'7F'21
788C 2340 00C5'CD
788F 2341
788F 2342 016C'CD
7892 2343
7892 2344 00'4F'21
7895 2345 00D1'CD
7898 2346
7898 2347 00'4F'21
789B 2348 00E9'CD
789E 2349
789E 2350 0139'CD
78A1 2351 010F'CD
78A4 2352
78A4 2353 40 3E
78A6 2354 30
78A7 2355 C0 3E
78A9 2356 30
78AA 2357
78AA 2358 00
78AB 2359 00
78AC 2360
78AC 2361 20
78AD 2362 10 E6
78AF 2363 008D'32
78B2 2364
78B2 2365 041D'CD
78B5 2366
78B5 2367
78B5 2368 0000'3A
78B8 2369 0F
78B9 2370 0853'DA
78BC 2371
78BC 2372 02 3E
78BE 2373 0000'32
78C1 2374
78C1 2375 00 3E
78C3 2376 008A'32
78C6 2377
78C6 2378 2D D3
78C8 2379
78C8 2380 00
78C9 2381 00
78CA 2382
78CA 2383 2C D3
78CC 2384

```

I 4

Fiche 1 Frame I4

Sequence 47

```

CALL CLR_OUTPUT      ;CLEAR THE OUTPUT BIT.
LXI H,ZERO_DAT        ;WRITE THE COUNTER INITIAL VALUE.
CALL WRITE_LOAD
LXI H,ONE_SHIFT+1     ;WRITE THE ALARM DATA.
CALL WRITE_ALARM
CALL LOAD_CNTR        ;LOAD AND ARM COUNTER 1.
CALL ARM_CNTR
CALL INC_CG_PNT       ;POINT TO COUNTER GROUP 2.
LXI H,TEST2B_DAT+4    ;WRITE THE MODE DATA.
CALL WRITE_MODE
CALL CLR_OUTPUT      ;CLEAR THE OUTPUT BIT.
LXI H,ZERO_DAT        ;WRITE THE COUNTER INITIAL VALUE.
CALL WRITE_LOAD
LXI H,ZERO_DAT        ;WRITE THE ALARM DATA.
CALL WRITE_ALARM
CALL LOAD_CNTR        ;LOAD AND ARM COUNTER 2.
CALL ARM_CNTR
MVI A,CLR_SER_OUT     ;CLEAR THE 8085 SERIAL OUTPUT.
SIM
MVI A,SET_SER_OUT     ;SET THE 8085 SERIAL OUTPUT TO
SIM                   ; ENABLE A TIMER INTERRUPT.
NOP                   ;ALLOW TIME FOR TIMER CONTROL
NOP                   ; SIGNALS TO SETTLE.
RIM                   ;READ RST 5.5 STATUS.
ANI <^X10>            ;MASK OUT STATUS BIT.
STA REC_DAT           ;STORE IN REC_DAT.
CALL CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
                        ; REPORT ERROR IF BYTE DOES NOT MATCH
LDA ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1$                 ;....JUMP TO BEGINNING OF ERROR LOOP.
MVI A,2
STA CON_ERR_NUM       ;SECOND PART OF TEST 2B.
MVI A,0
STA EXP_DAT           ;LOAD THE TIMER INTERRUPT ASSERTED
                        ; LOW DATA INTO EXP_DAT.
OUT SETTMR            ;START TIMER COUNTING.
NOP                   ;ALLOW TIME FOR TIMER INTERRUPT
NOP                   ; SIGNAL TO BECOME ASSERTED.
OUT CLRTMR            ;STOP TIMER.

```

```

ZZ-ENKBE-2.1 7000 4
78CC 2385 20 DB
78CE 2386 80 E6
78D0 2387 008D'32
78D3 2388
78D3 2389 041D'CD
78D6 2390
78D6 2391
78D6 2392 0000'3A
78D9 2393 0F
78DA 2394 085D'DA
78DD 2395
78DD 2396
78DD 2397 03 3E
78DF 2398 0000'32
78E2 2399
78E2 2400 10 3E
78E4 2401 008A'32
78E7 2402
78E7 2403 20
78E8 2404 10 E6
78EA 2405 008D'32
78ED 2406
78ED 2407 041D'CD
78F0 2408
78F0 2409
78F0 2410 0000'3A
78F3 2411 0F
78F4 2412 08E7'DA
78F7 2413
78F7 2414 04 3E
78F9 2415 0000'32
78FC 2416
78FC 2417 80 3E
78FE 2418 008A'32
7901 2419
7901 2420 40 3E
7903 2421 30
7904 2422
7904 2423 20 DB
7906 2424 80 E6
7908 2425 008D'32
790B 2426
790B 2427 041D'CD
790E 2428
790E 2429
790E 2430 0000'3A
7911 2431 0F
7912 2432 0901'DA
7915 2433
7915 2434 05 3E
7917 2435 0000'32
791A 2436
791A 2437 00 3E
791C 2438 008A'32
791F 2439
791F 2440 03 3E
7921 2441 000C'32
7924 2442 0C'81'21
7927 2443 00C5'CD
792A 2444

```

3\$:

4\$:

5\$:

J 4

```

IN      SUMREG      ;READ THE SUMMARY REGISTER.
ANI     <^X80>       ;SAVE THE MSB (TIMER INTERRUPT).
STA     REC_DAT      ;STORE IN REC_DAT.

CALL    CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
                                ; REPORT ERROR IF BYTE DOES NOT MATCH

LDA     ERROR_CON    ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC      2$           ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI     A,3          ;THIRD PART OF TEST 2B.
STA     CON_ERR_NUM

MVI     A,<^X10>      ;LOAD RST 5.5 REQUEST PENDING
STA     EXP_DAT       ; DATA INTO EXP_DAT.

RIM
ANI     <^X10>       ;READ RST 5.5 STATUS.
STA     REC_DAT       ;MASK OUT STATUS BIT.
                                ;STORE IN REC_DAT.

CALL    CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
                                ; REPORT ERROR IF BYTE DOES NOT MATCH

LDA     ERROR_CON    ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC      3$           ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI     A,4          ;FOURTH PART OF TEST 2B.
STA     CON_ERR_NUM

MVI     A,<^X80>      ;LOAD THE TIMER INTERRUPT UNASSERTED
STA     EXP_DAT       ; DATA INTO EXP_DAT.

MVI     A,CLR_SER_OUT ;CLEAR THE 8085 SERIAL OUTPUT.
SIM

IN      SUMREG      ;READ THE SUMMARY REGISTER.
ANI     <^X80>       ;SAVE THE MSB (TIMER INTERRUPT).
STA     REC_DAT      ;STORE IN REC_DAT.

CALL    CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
                                ; REPORT ERROR IF BYTE DOES NOT MATCH

LDA     ERROR_CON    ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC      4$           ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI     A,5          ;FITH PART OF TEST 2B.
STA     CON_ERR_NUM

MVI     A,0          ;LOAD THE POWERFAIL ASSERTED DATA
STA     EXP_DAT       ; INTO EXP_DAT.

MVI     A,3          ;LOAD THE MODE DATA INTO COUNTER
STA     CG_PNT        ; MODE REGISTER 3.
LXI     H,TEST2B_DAT+6
CALL    WRITE_MODE

```

Fiche 1 Frame J4 Sequence 48


```

ZZ-ENKBE-2.1 7000 4
792A 2445 00'51'21
792D 2446 00D1'CD
7930 2447
7930 2448 0124'CD
7933 2449 0139'CD
7936 2450
7936 2451 016C'CD
7939 2452
7939 2453 010F'CD
793C 2454
793C 2455 50 3E
793E 2456 017E'CD
7941 2457
7941 2458 20 DB
7943 2459 40 E6
7945 2460 008D'32
7948 2461
7948 2462 041D'CD
794B 2463
794B 2464
794B 2465 0000'3A
794E 2466 0F
794F 2467 091F'DA
7952 2468
7952 2469
7952 2470 06 3E
7954 2471 0000'32
7957 2472
7957 2473 40 3E
7959 2474 008A'32
795C 2475
795C 2476 0124'CD
795F 2477
795F 2478 016C'CD
7962 2479
7962 2480 20 DB
7964 2481 40 E6
7966 2482 008D'32
7969 2483
7969 2484 041D'CD
796C 2485
796C 2486
796C 2487 0000'3A
796F 2488 0F
7970 2489 095C'DA
7973 2490
7973 2491
7973 2492 3C D3
7975 2493
7975 2494
7975 2495
7975 2496
7975 2497
7975 2498
7975 2499
7975 2500
7975 2501
7975 2502
7975 2503
7975 2504

```

K 4

```

Fiche 1 Frame K4 Sequence 49
LXI H,ONE_SHIFT+1 ;LOAD THE INITIAL COUNT INTO THE
CALL WRITE_LOAD ; COUNTER 3 LOAD REGISTER.

CALL DISARM_CNTR ;DISARM AND LOAD THE COUNTER.
CALL LOAD_CNTR

CALL CLR_OUTPUT ;CLEAR THE OUTPUT BIT.

CALL ARM_CNTR ;ARM THE COUNTER.

MVI A,WAIT_CNT2 ;WAIT FOR A POWER FAIL SIGNAL.
CALL WAIT_LOOP

IN SUMREG ;READ THE SUMMARY REGISTER.
ANI <^X40> ;SAVE THE POWER FAIL INTERRUPT BIT.
STA REC_DAT ;STORE IN REC_DAT.

CALL CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
; REPORT ERROR IF BYTE DOES NOT MATCH

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 5$ ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI A,6 ;SIXTH PART OF TEST 2B.
STA CON_ERR_NUM

MVI A,<^X40> ;LOAD THE POWERFAIL UNASSERTED (HIGH)
STA EXP_DAT ; DATA INTO EXP_DAT.

CALL DISARM_CNTR ;DISARM COUNTER 3.

CALL CLR_OUTPUT ;CLEAR (UNASSERT) THE COUNTER 3 OUTPUT.

IN SUMREG ;READ THE SUMMARY REGISTER.
ANI <^X40> ;SAVE THE POWER FAIL INTERRUPT BIT.
STA REC_DAT ;STORE IN REC_DAT.

CALL CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
; REPORT ERROR IF BYTE DOES NOT MATCH

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 6$ ;...JUMP TO BEGINNING OF ERROR LOOP.

TAILR

```

6\$:

TEST2C

SUMMARY REGISTER TEST (USART INPUTS)

THE REMAINING SIX PINS OF THE SUMMARY REGISTER ARE CHECKED
BY SETTING AND CLEARING THE TRANSMIT AND RECEIVE READY SIGNALS FROM
EACH OF THE THREE USARTS. THE USARTS ARE PUT IN LOCAL LOOPBACK MODE

7975 2505
7975 2506
7975 2507
7975 2508
7975 2509
7975 2510
7975 2511
7975 2512
7975 2513
7975 2514
7975 2515
7975 2516
7975 2517
7975 2518
7975 2519
7975 2520
7975 2521
7975 2522
7975 2523
7975 2524
7975 2525
7975 2526
7975 2527
7975 2528
7975 2529
7975 2530
7975 2531
7975 2532
7975 2533
7975 2534
7975 2535
7975 2536
7975 2537
7975 2538
7975 2539
7975 2540
7975 2541
7975 2542
7975 2543
7975 2544
7975 2545
7975 2546
7975 2547
7975 2548
7975 2549
7975 2550
7975 2551
7975 2552
7975 2553
7975 2554
7975 2555
7975 2556
7975 2557
7975 2558
7975 2559
7975 2560
7975 2561
7975 2562
7975 2563
7975 2564

TO MANIPULATE THESE SIGNALS. SUMMARY REGISTER BITS 1, 5, 0 AND 3 ARE NOT TESTED IF REMOTE DIAGNOSTICS ARE BEING RUN. BITS 1 AND 5 ARE NOT TESTED IF APT IS BEING RUN.

TEST STEPS:

TERMINAL RECEIVE BIT (SUM REG BIT 3)

- 1) PUT TERMINAL USART IN LOOPBACK MODE.
- 2) WRITE DATA TO THE TERMINAL USART DATA BUFFER.
- 3) WAIT FOR A RECEIVE DONE STATUS REG SIGNAL.
- 4) READ SUMMARY REGISTER AND CHECK FOR BIT 3 CLEAR.
- 5) READ THE TERMINAL DATA BUFFER (TO LEAVE IT EMPTY).
- 6) RESTORE TERMINAL USART TO NORMAL MODE.
- 7) PRINT ERROR IF ANY.
- 8) PUT TERMINAL USART IN LOOPBACK MODE.
- 9) WRITE DATA TO THE TERMINAL USART DATA BUFFER.
- 10) WAIT FOR A RECEIVE DONE STATUS REG SIGNAL.
- 11) READ DATA FROM THE TERMINAL USART DATA BUFFER.
- 12) READ SUMMARY REGISTER AND CHECK FOR BIT 3 SET.
- 13) RESTORE TERMINAL USART TO NORMAL MODE.
- 14) PRINT ERROR IF ANY.

TERMINAL TRANSMIT BIT (SUM REG BIT 0)

- 1) SET TERMINAL USART TO LOOPBACK MODE.
- 2) WRITE DATA TO THE TERMINAL USART DATA BUFFER.
- 3) READ THE SUMMARY REGISTER AND CHECK FOR BIT 0 SET.
- 4) WAIT FOR A RECEIVED DONE STATUS REGISTER SIGNAL.
- 5) READ THE TERMINAL DATA BUFFER (TO LEAVE IT EMPTY).
- 6) RESTORE TERMINAL USART TO NORMAL MODE.
- 7) PRINT ERROR IF ANY.
- 8) SET TERMINAL USART TO LOOPBACK MODE.
- 9) WRITE DATA TO THE TERMINAL USART DATA BUFFER.
- 10) WAIT FOR A TRANSMIT READY STATUS REG SIGNAL.
- 11) READ THE SUMMARY REGISTER AND CHECK FOR BIT 0 CLEAR.
- 12) WAIT FOR A RECEIVED DONE STATUS REGISTER SIGNAL.
- 13) READ THE TERMINAL DATA BUFFER (TO LEAVE IT EMPTY).
- 14) RESTORE USARTS TO NORMAL MODE.
- 15) PRINT ERROR IF ANY.

REPEAT ABOVE STEPS FOR TU58 AND REMOTE USARTS.

ASSUMPTIONS: ALL PREVIOUS TESTS (INCLUDING ROM SELF TESTS) HAVE RUN SUCCESSFULLY.

ERROR1: SUMMARY REGISTER BIT 3 (TERMINAL RECEIVE BIT) ERROR.
ERROR2: SUMMARY REGISTER BIT 0 (TERMINAL TRANSMIT BIT) ERROR.
ERROR3: SUMMARY REGISTER BIT 4 (TU58 RECEIVE BIT) ERROR.
ERROR4: SUMMARY REGISTER BIT 2 (TU58 TRANSMIT BIT) ERROR.
ERROR5: SUMMARY REGISTER BIT 5 (REMOTE RECEIVE BIT) ERROR.
ERROR6: SUMMARY REGISTER BIT 1 (REMOTE TRANSMIT BIT) ERROR.

DEBUG: 1) CHECK EACH USART (E5, E6 AND E7) 'RX RDY L' FOR LOW VALUE AFTER SENDING DATA TO THE USART.
2) CHECK CORRESPONDING PINS ON SUMMARY REGISTER (E23 AND

22-ENKBE-2.1 7000 4

7975 2565
7975 2566
7975 2567
7975 2568
7975 2569
7975 2570
7975 2571
7975 2572
7975 2573
7975 2574
7975 2575
7975 2576
7975 2577
7975 2578
7975 2579
7975 2580
7975 2581
7975 2582
7975 2583
7975 2584 0000'CD 2C 3E
OF 0000'3A
C9 0982'D2
OF 0000'3A
3D D3 0BA5'DA

798B 2585
798B 2586 01EF'CD
798E 2587
798E 2588 OF 000A'3A
0A42'DA
7995 2589
7995 2590
7995 2591 01 3E
7997 2592 0000'32
799A 2593
799A 2594 00 3E
799C 2595 008A'32
799F 2596
799F 2597 019E'CD
79A2 2598
79A2 2599 48 D3
79A4 2600
79A4 2601 49 DB
79A6 2602 02 E6
79A8 2603 09A4'CA
79AB 2604
79AB 2605 20 DB
79AD 2606 08 E6
79AF 2607 008D'32
79B2 2608
79B2 2609 48 DB
79B4 2610
79B4 2611 01D7'CD
79B7 2612
79B7 2613 041D'CD
79BA 2614
79BA 2615
79BA 2616 0000'3A
79BD 2617 OF
79BE 2618 099F'DA
79C1 2619

M 4

Fiche 1 Frame M4

Sequence 51

- E24) FOR LOW VALUES AFTER THE DATA WRITE.
3) CHECK EACH USART 'RX RDY L' FOR HIGH VALUE AFTER THE DATA READ.
4) CHECK CORRESPONDING PINS ON THE SUMMARY REGISTER FOR HIGH VALUES AFTER THE DATA READ.
5) CHECK FOR 'TX RDY L' FOR HIGH VALUE (PULSE) AFTER DATA WRITE.
6) CHECK CORRESPONDING BITS ON SUMMARY REGISTER FOR HIGH VALUES (PULSE) AFTER DATA WRITE.
7) CHECK FOR 'TX RDY L' FOR LOW VALUE (WILL GO LOW AFTER A DELAY) AFTER DATA WRITE.
8) CHECK CORRESPONDING BITS ON THE SUMMARY REGISTER FOR LOW VALUES (AGAIN, THE DELAY) AFTER DATA WRITE.

TEST2C:

HEAD <^X2C>

CALL CHECK_REMOTE ;CHECK TO SEE IF UNDER RD.
IFN SKIP_UA1 ;SKIP TERMINAL USART TEST IF FLAG
; IS ASSERTED.
MVI A,1 ;TERMINAL USART RECEIVE BIT BEING
STA CON_ERR_NUM ; TESTED.
MVI A,0 ;TERMINAL RECEIVE BIT ASSERTED.
STA EXP_DAT
1\$: CALL SET_TT_LB ;SET TERMINAL USART TO LOOPBACK MODE.
OUT TTDB ;WRITE DATA TO TERMINAL USART.
2\$: IN TTSR ;WAIT FOR A RECEIVED DONE.
ANI REC_DONE
JZ 2\$
IN SUMREG ;READ THE SUMMARY REGISTER.
ANI TER_REC_BIT ;SAVE BIT 3.
STA REC_DAT ;STORE IN REC_DAT.
IN TTDB ;CLEAR THE TERMINAL USART DATA BUFFER.
CALL RESTORE_TT ;RESTORE TERMINAL USART TO NORMAL MODE.
CALL CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
; REPORT ERROR IF BYTE DOES NOT MATCH
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1\$;...JUMP TO BEGINNING OF ERROR LOOP.

79C1	2620	
79C1	2621	
79C1	2622	08 3E
79C3	2623	008A'32
79C6	2624	
79C6	2625	019E'CD
79C9	2626	
79C9	2627	48 D3
79CB	2628	
79CB	2629	49 DB
79CD	2630	02 E6
79CF	2631	09CB'CA
79D2	2632	
79D2	2633	48 DB
79D4	2634	
79D4	2635	20 DB
79D6	2636	08 E6
79D8	2637	008D'32
79DB	2638	
79DB	2639	01D7'CD
79DE	2640	
79DE	2641	041D'CD
79E1	2642	
79E1	2643	
79E1	2644	0000'3A
79E4	2645	0F
79E5	2646	09C6'DA
79E8	2647	
79E8	2648	
79E8	2649	
79E8	2650	02 3E
79EA	2651	0000'32
79ED	2652	
79ED	2653	01 3E
79EF	2654	008A'32
79F2	2655	
79F2	2656	019E'CD
79F5	2657	
79F5	2658	48 D3
79F7	2659	
79F7	2660	20 DB
79F9	2661	01 E6
79FB	2662	008D'32
79FE	2663	
79FE	2664	49 DB
7A00	2665	02 E6
7A02	2666	09FE'CA
7A05	2667	
7A05	2668	48 DB
7A07	2669	
7A07	2670	01D7'CD
7A0A	2671	
7A0A	2672	041D'CD
7A0D	2673	
7A0D	2674	
7A0D	2675	0000'3A
7A10	2676	0F
7A11	2677	09F2'DA
7A14	2678	
7A14	2679	

Sequence 52

```

MVI    A,TER_REC_BIT    ;TERMINAL USART RECEIVE BIT UNASSERTED.
STA    EXP_DAT

CALL    SET_TT_LB        ;SET TERMINAL USART TO LOOPBACK MODE.

OUT     TTDB             ;WRITE DATA TO TERMINAL USART.

IN      TTSR             ;WAIT FOR A RECEIVED DONE.
ANI     REC_DONE
JZ      4$

IN      TTDB             ;READ DATA FROM THE USART.

IN      SUMREG           ;READ THE SUMMARY REGISTER.
ANI     TER_REC_BIT      ;SAVE BIT 3.
STA     REC_DAT          ;STORE IN REC_DAT.

CALL    RESTORE_TT       ;RESTORE TERMINAL USART TO NORMAL MODE.

CALL    CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
                        ; REPORT ERROR IF BYTE DOES NOT MATCH

LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC      3$               ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI     A,2              ;TERMINAL USART TRANSMIT BIT BEING
STA     CON_ERR_NUM      ; TESTED.

MVI     A,TER_TX_BIT     ;TERMINAL USART TRANSMIT BIT
STA     EXP_DAT          ; UNASSERTED.

CALL    SET_TT_LB        ;SET TERMINAL USART TO LOOPBACK MODE.

OUT     TTDB             ;WRITE DATA TO TERMINAL USART.

IN      SUMREG           ;READ THE SUMMARY REGISTER.
ANI     TER_TX_BIT       ;SAVE BIT 0.
STA     REC_DAT          ;STORE IN REC_DAT.

IN      TTSR             ;WAIT FOR A RECEIVED DONE.
ANI     REC_DONE
JZ      6$

IN      TTDB             ;CLEAR THE TERMINAL USART DATA BUFFER.

CALL    RESTORE_TT       ;RESTORE TERMINAL USART TO NORMAL MODE.

CALL    CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
                        ; REPORT ERROR IF BYTE DOES NOT MATCH

LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC      5$               ;...JUMP TO BEGINNING OF ERROR LOOP.

```


7A14	2680		
7A14	2681	00	3E
7A16	2682	008A'	32
7A19	2683		
7A19	2684	019E'	CD
7A1C	2685		
7A1C	2686	48	D3
7A1E	2687		
7A1E	2688	49	DB
7A20	2689	01	E6
7A22	2690	0A1E'	CA
7A25	2691		
7A25	2692	20	DB
7A27	2693	01	E6
7A29	2694	008D'	32
7A2C	2695		
7A2C	2696	49	DB
7A2E	2697	02	E6
7A30	2698	0A2C'	CA
7A33	2699		
7A33	2700	48	DB
7A35	2701		
7A35	2702	01D7'	CD
7A38	2703		
7A38	2704	041D'	CD
7A3B	2705		
7A3B	2706		
7A3B	2707	0000'	3A
7A3E	2708	0F	
7A3F	2709	0A19'	DA
7A42	2710		
7A42	2711		
7A42	2712		
7A42	2713		
7A42	2714	03	3E
7A44	2715	0000'	32
7A47	2716		
7A47	2717	00	3E
7A49	2718	008A'	32
7A4C	2719		
7A4C	2720		
7A4C	2721	01BB'	CD
7A4F	2722		
7A4F	2723	44	D3
7A51	2724		
7A51	2725	45	DB
7A53	2726	02	E6
7A55	2727	0A51'	CA
7A58	2728		
7A58	2729	20	DB
7A5A	2730	10	E6
7A5C	2731	008D'	32
7A5F	2732		
7A5F	2733	44	DB
7A61	2734		
7A61	2735	01DF'	CD
7A64	2736		
7A64	2737	041D'	CD
7A67	2738		
7A67	2739		

85

Fiche 1 Frame B5

Sequence 53

[illegible]

ZZ
.M
Ps

PS

CP

Ph

In

Pa

sy
PaSY
PSCF
AS

Th

28
Th52
15

1

Ma

SY

0

Th

/S

100

```

ZZ-ENKBE-2.1 7000 4
7A67 2740 0000'3A
7A6A 2741 0F
7A6B 2742 0A4C'DA
7A6E 2743
7A6E 2744
7A6E 2745
7A6E 2746 10 3E
7A70 2747 008A'32
7A73 2748
7A73 2749 01BB'CD
7A76 2750
7A76 2751 44 D3
7A78 2752
7A78 2753 45 DB
7A7A 2754 02 E6
7A7C 2755 0A78'CA
7A7F 2756
7A7F 2757 44 DB
7A81 2758
7A81 2759 20 DB
7A83 2760 10 E6
7A85 2761 008D'32
7A88 2762
7A88 2763 01DF'CD
7A8B 2764
7A8B 2765 041D'CD
7A8E 2766
7A8E 2767
7A8E 2768 0000'3A
7A91 2769 0F
7A92 2770 0A73'DA
7A95 2771
7A95 2772
7A95 2773
7A95 2774 04 3E
7A97 2775 0000'32
7A9A 2776
7A9A 2777 04 3E
7A9C 2778 008A'32
7A9F 2779
7A9F 2780 01BB'CD
7AA2 2781
7AA2 2782 44 D3
7AA4 2783
7AA4 2784 20 DB
7AA6 2785 04 E6
7AA8 2786 008D'32
7AAB 2787
7AAB 2788 45 DB
7AAD 2789 02 E6
7AAF 2790 0AAB'CA
7AB2 2791
7AB2 2792 44 DB
7AB4 2793
7AB4 2794 01DF'CD
7AB7 2795
7AB7 2796 041D'CD
7ABA 2797
7ABA 2798
7ABA 2799 0000'3A

```

C 5

```

LDA ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 11$             ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI A,TU58_REC_BIT ;TU58 USART RECEIVE BIT UNASSERTED.
STA EXP_DAT

CALL SET_TU_LB      ;SET TU58 USART TO LOOPBACK MODE.
OUT TUDB             ;WRITE DATA TO TU58 USART.

IN TUSR              ;WAIT FOR A RECEIVED DONE.
ANI REC_DONE
JZ 14$

IN TUDB              ;READ DATA FROM THE TU58 USART.

IN SUMREG            ;READ THE SUMMARY REGISTER.
ANI TU58_REC_BIT     ;SAVE BIT 4.
STA REC_DAT          ;STORE IN REC_DAT.

CALL RESTORE_TU      ;RESTORE TU58 USART TO NORMAL MODE.

CALL CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
                        ; REPORT ERROR IF BYTE DOES NOT MATCH

LDA ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 13$             ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI A,4              ;TU58 USART TRANSMIT BIT BEING TESTED.
STA CON_ERR_NUM

MVI A,TU58_TX_BIT    ;TU58 USART TRANSMIT BIT UNASSERTED.
STA EXP_DAT

CALL SET_TU_LB      ;SET TU58 USART TO LOOPBACK MODE.
OUT TUDB             ;WRITE DATA TO TU58 USART.

IN SUMREG            ;READ THE SUMMARY REGISTER.
ANI TU58_TX_BIT     ;SAVE BIT 2.
STA REC_DAT          ;STORE IN REC_DAT.

IN TUSR              ;WAIT FOR A RECEIVED DONE.
ANI REC_DONE
JZ 16$

IN TUDB              ;CLEAR THE TU58 USART DATA BUFFER.

CALL RESTORE_TU      ;RESTORE TU58 USART TO NORMAL MODE.

CALL CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
                        ; REPORT ERROR IF BYTE DOES NOT MATCH

LDA ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...

```

Fiche 1 Frame C5 Sequence 54

ZZ

Fi

ZZ-ENKBE-2.1 7000 4

7ABD 2800 0F
 7ABE 2801 0A9F'DA
 7AC1 2802
 7AC1 2803
 7AC1 2804
 7AC1 2805 00 3E
 7AC3 2806 008A'32
 7AC6 2807
 7AC6 2808 01BB'CD
 7AC9 2809
 7AC9 2810 44 D3
 7ACB 2811
 7ACB 2812 45 DB
 7ACD 2813 01 E6
 7ACF 2814 0ACB'CA
 7AD2 2815
 7AD2 2816 20 DB
 7AD4 2817 04 E6
 7AD6 2818 008D'32
 7AD9 2819
 7AD9 2820 45 DB
 7ADB 2821 02 E6
 7ADD 2822 0AD9'CA
 7AE0 2823
 7AE0 2824 44 DB
 7AE2 2825
 7AE2 2826 01DF'CD
 7AE5 2827
 7AE5 2828 041D'CD
 7AE8 2829
 7AE8 2830
 7AE8 2831 0000'3A
 7AEB 2832 0F
 7AEC 2833 0AC6'DA
 7AEF 2834
 7AEF 2835
 7AEF 2836
 7AEF 2837
 7AEF 2838
 7AEF 2839
 7AEF 2840 0F 000B'3A
 7AEF 2840 0BA3'DA
 7AF6 2841
 7AF6 2842
 7AF6 2843 05 3E
 7AF8 2844 0000'32
 7AFB 2845
 7AFB 2846 00 3E
 7AFD 2847 008A'32
 7B00 2848
 7B00 2849 01C9'CD
 7B03 2850
 7B03 2851 40 D3
 7B05 2852
 7B05 2853 41 DB
 7B07 2854 02 E6
 7B09 2855 0B05'CA
 7B0C 2856
 7B0C 2857 20 DB
 7B0E 2858 20 E6

 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *

17\$:

18\$:

19\$:

21\$:

22\$:

D 5

RRC
 JC

15\$

MVI
 STA

A,0
 EXP_DAT

CALL

SET_TU_LB

OUT

TUDB

IN
 ANI
 JZ

TUSR
 TX_RDY
 18\$

IN
 ANI
 STA

SUMREG
 TU58_TX_BIT
 REC_DAT

IN
 ANI
 JZ

TUSR
 REC_DONE
 19\$

IN

TUDB

CALL

RESTORE_TU

CALL

CHECK_2B_2C_ERROR

LDA
 RRC
 JC

ERROR_CON
 17\$

IFN

SKIP_UA3

MVI
 STA

A,5
 CON_ERR_NUM

MVI
 STA

A,0
 EXP_DAT

CALL

SET_TR_LB

OUT

TRDB

IN
 ANI
 JZ

TRSR
 REC_DONE
 22\$

IN
 ANI

SUMREG
 REM_REC_BIT

Fiche 1 Frame D5

Sequence 55

;...JUMP TO BEGINNING OF ERROR LOOP.

;TU58 USART TRANSMIT BIT ASSERTED.

;SET TU58 USART TO LOOPBACK MODE.

;WRITE DATA TO TU58 USART.

;WAIT FOR A TRANSMIT READY IN THE
 ; STATUS REGISTER.

;READ THE SUMMARY REGISTER.

;SAVE BIT 2.

;STORE IN REC_DAT.

;WAIT FOR A RECEIVED DONE.

;CLEAR THE TU58 USART DATA BUFFER.

;RESTORE TU58 USART TO NORMAL MODE.

;COMPARE EXPECTED AND RECEIVED DATA
 ; REPORT ERROR IF BYTE DOES NOT MATCH

;IF THE ERROR LOOPING FLAG IS SET...

;...JUMP TO BEGINNING OF ERROR LOOP.

;SKIP LAST USART TEST IF USART3_FLAG

; IS SET.

;REMOTE USART RECEIVE BIT BEING TESTED.

;REMOTE RECEIVE BIT ASSERTED.

;SET REMOTE USART TO LOOPBACK MODE.

;WRITE DATA TO REMOTE USART.

;WAIT FOR A RECEIVED DONE.

;READ THE SUMMARY REGISTER.

;SAVE BIT 5.

22-ENKBE-2.1 7000 4

```

7B10 2859 008D'32 *
7B13 2860 *
7B13 2861 40 DB *
7B15 2862 *
7B15 2863 01E7'CD *
7B18 2864 *
7B18 2865 041D'CD *
7B1B 2866 *
7B1B 2867 *
7B1B 2868 0000'3A *
7B1E 2869 0F *
7B1F 2870 0B00'DA *
7B22 2871 *
7B22 2872 *
7B22 2873 *
7B22 2874 20 3E *
7B24 2875 008A'32 *
7B27 2876 *
7B27 2877 01C9'CD *
7B2A 2878 *
7B2A 2879 40 D3 *
7B2C 2880 *
7B2C 2881 41 DB *
7B2E 2882 02 E6 *
7B30 2883 0B2C'CA *
7B33 2884 *
7B33 2885 40 DB *
7B35 2886 *
7B35 2887 20 DB *
7B37 2888 20 E6 *
7B39 2889 008D'32 *
7B3C 2890 *
7B3C 2891 01E7'CD *
7B3F 2892 *
7B3F 2893 041D'CD *
7B42 2894 *
7B42 2895 *
7B42 2896 0000'3A *
7B45 2897 0F *
7B46 2898 0B27'DA *
7B49 2899 *
7B49 2900 *
7B49 2901 *
7B49 2902 06 3E *
7B4B 2903 0000'32 *
7B4E 2904 *
7B4E 2905 02 3E *
7B50 2906 008A'32 *
7B53 2907 *
7B53 2908 01C9'CD *
7B56 2909 *
7B56 2910 40 D3 *
7B58 2911 *
7B58 2912 20 DB *
7B5A 2913 02 E6 *
7B5C 2914 008D'32 *
7B5F 2915 *
7B5F 2916 41 DB *
7B61 2917 02 E6 *
7B63 2918 0B5F'CA *

```

E 5

Fiche 1 Frame E5 Sequence 56

```

STA REC_DAT ;STORE IN REC_DAT.
IN TRDB ;CLEAR THE REMOTE USART DATA BUFFER.
CALL RESTORE_TR ;RESTORE REMOTE USART TO NORMAL.
CALL CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
; REPORT ERROR IF BYTE DOES NOT MATCH
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 21$ ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI A,REM_REC_BIT ;REMOTE USART RECEIVE BIT UNASSERTED.
STA EXP_DAT
CALL SET_TR_LB ;SET REMOTE USART TO LOOPBACK MODE.
OUT TRDB ;WRITE DATA TO REMOTE USART.
IN TRSR ;WAIT FOR A RECEIVED DONE.
ANI REC_DONE
JZ 24$

IN TRDB ;READ DATA FROM THE USART.
IN SUMREG ;READ THE SUMMARY REGISTER.
ANI REM_REC_BIT ;SAVE BIT 3.
STA REC_DAT ;STORE IN REC_DAT.
CALL RESTORE_TR ;RESTORE REMOTE USART TO NORMAL.
CALL CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
; REPORT ERROR IF BYTE DOES NOT MATCH
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 23$ ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI A,6 ;REMOTE USART TRANSMIT BIT BEING
STA CON_ERR_NUM ;TESTED.
MVI A,REM_TX_BIT ;REMOTE USART TRANSMIT BIT UNASSERTED.
STA EXP_DAT
CALL SET_TR_LB ;SET REMOTE USART TO LOOPBACK MODE.
OUT TRDB ;WRITE DATA TO REMOTE USART.
IN SUMREG ;READ THE SUMMARY REGISTER.
ANI REM_TX_BIT ;SAVE BIT 2.
STA REC_DAT ;STORE IN REC_DAT.
IN TRSR ;WAIT FOR A RECEIVED DONE.
ANI REC_DONE
JZ 26$

```

23\$:

24\$:

25\$:

26\$:

7B66	2919	
7B66	2920	40 DB
7B68	2921	
7B68	2922	01E7'CD
7B6B	2923	
7B6B	2924	041D'CD
7B6E	2925	
7B6E	2926	
7B6E	2927	0000'3A
7B71	2928	0F
7B72	2929	0B53'DA
7B75	2930	
7B75	2931	
7B75	2932	
7B75	2933	00 3E
7B77	2934	008A'32
7B7A	2935	
7B7A	2936	01C9'CD
7B7D	2937	
7B7D	2938	40 D3
7B7F	2939	
7B7F	2940	41 DB
7B81	2941	01 E6
7B83	2942	0B7F'CA
7B86	2943	
7B86	2944	20 DB
7B88	2945	02 E6
7B8A	2946	008D'32
7B8D	2947	
7B8D	2948	41 DB
7B8F	2949	02 E6
7B91	2950	0B8D'CA
7B94	2951	
7B94	2952	40 DB
7B96	2953	
7B96	2954	01E7'CD
7B99	2955	
7B99	2956	041D'CD
7B9C	2957	
7B9C	2958	
7B9C	2959	0000'3A
7B9F	2960	0F
7BA0	2961	0B7A'DA
7BA3	2962	
7BA3	2963	
7BA3	2964	
7BA3	2965	
7BA3	2966	3C D3
7BA5	2967	
7BA5	2968	
7BA5	2969	
7BA5	2970	
7BA5	2971	
7BA5	2972	
7BA5	2973	
7BA5	2974	
7BA5	2975	
7BA5	2976	
7BA5	2977	
7BA5	2978	

Sequence 57

```

IN      TRDB      ;CLEAR THE REMOTE USART DATA BUFFER.
CALL    RESTORE_TR ;RESTORE REMOTE USART TO NORMAL.
CALL    CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
                                ; REPORT ERROR IF BYTE DOES NOT MATCH
LDA      ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC       25$      ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI      A,0      ;REMOTE USART TRANSMIT BIT ASSERTED.
STA      EXP_DAT

CALL     SET_TR_LB ;SET REMOTE USART TO LOOPBACK MODE.
OUT      TRDB      ;WRITE DATA TO REMOTE USART.

IN       TRSR      ;WAIT FOR A TRANSMIT READY IN THE
ANI      TX_RDY    ; STATUS REGISTER.
JZ       28$

IN       SUMREG     ;READ THE SUMMARY REGISTER.
ANI      REM_TX_BIT ;SAVE BIT 2.
STA      REC_DAT    ;STORE IN REC_DAT.

IN       TRSR      ;WAIT FOR A RECEIVED DONE.
ANI      REC_DONE
JZ       29$

IN       TRDB      ;CLEAR THE REMOTE USART DATA BUFFER.
CALL     RESTORE_TR ;RESTORE THE REMOTE USART TO NORMAL.
CALL     CHECK_2B_2C_ERROR ;COMPARE EXPECTED AND RECEIVED DATA
                                ; REPORT ERROR IF BYTE DOES NOT MATCH
LDA      ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC       27$      ;...JUMP TO BEGINNING OF ERROR LOOP.

ENDIF

```

TAIL

TEST2D

INCREMENT THROUGH WCS TEST

INC WR[0] INSTRUCTIONS ARE LOADED INTO EVERY WCS LOCATION EXCEPT THE LAST TWO, WHICH ARE LOADED WITH SET CPU ATTN AND JUMP SELF INSTRUCTIONS. THE UPC IS LOADED WITH ZERO AND THE CLOCK IS STARTED. WHEN THE CPU ATTN SIGNAL IS RECEIVED BY THE CONSOL, THE

22-ENKBE-2.1 7000 4

7BA5 2979
7BA5 2980
7BA5 2981
7BA5 2982
7BA5 2983
7BA5 2984
7BA5 2985
7BA5 2986
7BA5 2987
7BA5 2988
7BA5 2989
7BA5 2990
7BA5 2991
7BA5 2992
7BA5 2993
7BA5 2994
7BA5 2995
7BA5 2996
7BA5 2997
7BA5 2998
7BA5 2999
7BA5 3000
7BA5 3001
7BA5 3002
7BA5 3003
7BA5 3004
7BA5 3005
7BA5 3006
7BA5 3007
7BA5 3008
7BA5 3009
7BA5 3010
7BA5 3011
7BA5 3012
7BA5 3013
7BA5 3014
7BA5 3015
7BA5 3016
7BA5 3017 2D 3E
7BA7 3018
7BA7 3019 0000'CD
7BAA 3020
7BAA 3021 0F 0000'3A
0BB2'D2
7BB1 3022 C9
7BB2 3023
7BB2 3024
7BB2 3025 0F 0000'3A
0BBC'D2
7BB9 3026 0D33'C3
7BBC 3027
7BBC 3028
7BBC 3029 3D D3
7BBE 3030
7BBE 3031 0BC4'C3
7BC1 3032
7BC1 3033 0CEE'C3
7BC4 3034
7BC4 3035 008F'CD
7BC7 3036

G 5

Fiche 1 Frame G5 Sequence 58
CONTENTS OF WR[0] ARE CHECKED FOR A VALUE EQUAL TO WCS_SIZE-1.

TEST STEPS:

- 1) SIZE WCS.
- 2) WRITE INCR WR[0] INSTRUCTIONS TO WCS LOCATIONS.
- 3) WRITE SET CPU ATTN TO SECOND TO LAST WCS LOCATION.
- 4) WRITE JUMP[*] TO LAST LOCATION.
- 5) PREPARE TO RETURN FROM PARITY ERROR.
- 5) CLEAR WR[0].
- 6) CLEAR THE UPC.
- 7) START THE CPU CLOCK.
- 8) WAIT FOR CPU ATTN, TIMEOUT OR PARITY ERROR TRAP.
- 9) STOP CLOCK AND COMPARE CONTENTS OF WR[0] TO WCS_SIZE-2.
- 10) PRINT ERROR IF ANY.

ASSUMPTIONS: PREVIOUS TESTS HAVE RUN SUCCESSFULLY.

ERROR1: CPU TIMEOUT.

ERROR2: INCREMENT THROUGH WCS TEST ERROR.

ERROR3: PARITY ERROR OCCURRED SOMEWHERE IN WCS.

NOTE: EXPECTED DATA IS THE VALUE OF WCS_SIZE-1, RECEIVED DATA IS
IS THE VALUE RETURNED FROM WRO AND ADDITIONAL DATA CONTAINS
THE UPC CONTENTS.

DEBUG: 1) IF ALL THE PREVIOUS TESTS HAVE RUN SUCCESSFULLY, THEN
THIS TEST SHOULD THEORETICALLY DO THE SAME. HOWEVER, THIS
TEST IS USED TO DETERMINE IF CODE CAN BE RUN FROM WCS AT
SPEED AND THEREFORE MAY FIND ERRORS IN DOING SO.

```
*****
TEST2D:      MVI      A,<^X2D>      ;TEST 2D INDICATOR.
              CALL     CON_BEGIN_TEST ;REPORT BEGINNING OF TEST.
              IF       EOS_FLG       ;END OF SECTION.
              RET
              ENDIF
              IF       SKIP_TEST      ;IS THE TEST TO BE SKIPPED.
              JMP      END_TEST2D
              ENDIF
              OUT      SETLIT         ;TURN ON RUN LIGHT.
              JMP      TEST2D_PAR_ERR+3 ;SKIP NEXT INSTRUCTION.
TEST2D_PAR_ERR: JMP      6$           ;RETURN FROM A PARITY ERROR.
              CALL     CHECK_RD       ;CALL BOOT CODE IF RD IS IN CONTROL
```



```

ZZ-ENKBE-2.1 7000 4
7BC7 3037 0000'CD
7BCA 3038
7BCA 3039 008F'CD
7BCD 3040
7BCD 3041 0212'CD
7BD0 3042
7BD0 3043 029F'CD
7BD3 3044
7BD3 3045 0016'2A
7BD6 3046 008A'22
7BD9 3047 00'8A'21
7BDC 3048 0000'CD
7BDF 3049
7BDF 3050 00'43'21
7BE2 3051 00'00'11
7BE5 3052 0000'CD
7BE8 3053
7BE8 3054 00'00'21
7BEB 3055 7E
7BEC 3056 0A D3
7BEE 3057 23
7BEF 3058 7E
7BF0 3059 09 D3
7BF2 3060 23
7BF3 3061 7E
7BF4 3062 08 D3
7BF6 3063
7BF6 3064 00 00 21
7BF9 3065 0000'22
7BFC 3066 0000'CD
7BFF 3067
7BFF 3068 008F'CD
7C02 3069
7C02 3070 37 D3
7C04 3071 36 D3
7C06 3072
7C06 3073 00'00'21
                                02 0E 00'16'11
                                0C21'CA 0000'CD
7C14 3074
7C14 3075
7C14 3076 00'00'21
7C17 3077 0000'CD
7C1A 3078
7C1A 3079 27 D3
7C1C 3080 26 D3
7C1E 3081
7C1E 3082 0BFF'C3
7C21 3083
7C21 3084
7C21 3085
7C21 3086 00'46'21
7C24 3087 00'00'11
7C27 3088 0000'CD
7C2A 3089 0016'2A
7C2D 3090 0000'22
7C30 3091 00'00'21
7C33 3092 0000'CD
7C36 3093 0000'CD
7C39 3094

```

```

*****
*
*
*
*
*
*
*
*
*
*
*
*****

```

1\$:

2\$:

```

H 5
Fiche 1 Frame H5 Sequence 59
CALL CLEAR_CPU_ATT ;CLEAR CPU ATTN.
CALL CHECK_RD ;CALL BOOT CODE IF RD IS IN CONTROL
CALL WCS_SIZER ;SIZE WCS.
CALL WCS_SIZE_P ;PRINT THE WCS SIZE.
LHLD WCS_SIZE ;DECREMENT WCS SIZE BY ONE AND
SHLD EXP_DAT ; STORE IN EXP_DAT.
LXI H,EXP_DAT
CALL DECR_WORD

LXI H,INC_WRO ;MOVE AN INCREMENT WRO INSTRUCTION
LXI D,WCS_VALUE ; INTO WCS VALUE.
CALL MOVER_3

LXI H,WCS_VALUE ;LOAD THE INSTRUCTION INTO THE WCS
MOV A,M ; LOAD REGISTERS.
OUT WCSB2
INX H
MOV A,M
OUT WCSB1
INX H
MOV A,M
OUT WCSB0

LXI H,0 ;LOAD THE UPC WITH ZERO.
SHLD WCS_ADDRESS
CALL LD_OPC

CALL CHECK_RD ;CALL BOOT CODE IF RD IS IN CONTROL
OUT SETWCK ;LOAD THE INCREMENT INSTRUCTION
OUT CLRWCK ; INTO WCS.

IFNEQ WCS_ADDRESS,WCS_SIZE,2 ;HAS THE INSTRUCTION BEEN

                                ; LOADED INTO ALL OF WCS.

LXI H,WCS_ADDRESS ;POINT TO THE NEXT WCS LOCATION.
CALL INC_WORD

OUT SETUPK ;INCREMENT THE UPC TOO.
OUT CLRUPK

JMP 2$ ;REPEAT FOR THE NEXT WCS LOCATION.

ENDIF

LXI H,SET_CPU_ATT ;LOAD A SET CPU ATTN INSTRUCTION
LXI D,WCS_VALUE ; INTO THE SECOND TO LAST WCS
CALL MOVER_3 ; LOCATION.
LHLD WCS_SIZE
SHLD WCS_ADDRESS
LXI H,WCS_ADDRESS
CALL DECR_WORD
CALL WCS_WRITE

```

```

ZZ-ENKBE-2.1 7000 4
7C39 3095 0333'CD
7C3C 3096
7C3C 3097
7C3C 3098 008F'CD
7C3F 3099
7C3F 3100 00'4C'21
7C42 3101 00'00'11
7C45 3102 0000'CD
7C48 3103 0016'2A
7C4B 3104 0000'22
7C4E 3105 0000'32 3C AF
7C53 3106 0000'CD
7C56 3107
7C56 3108 00'83'21
7C59 3109 4C 22 11
7C5C 3110 0000'CD
7C5F 3111
7C5F 3112 008F'CD
7C62 3113
7C62 3114 00'01'21
7C65 3115 02F8'CD
7C68 3116
7C68 3117 0000'CD
7C6B 3118
7C6B 3119 00 00 21
7C6E 3120 0000'22
7C71 3121 0000'CD
7C74 3122
7C74 3123 A0 D3
7C76 3124
7C76 3125 21 D3
7C78 3126
7C78 3127
7C78 3128 0009'32 AF
7C7C 3129
7C7C 3130 FF 3E
7C7E 3131 0010'32
7C81 3132
7C81 3133 008F'CD
7C84 3134
7C84 3135 83 DB 3$:
7C86 3136 1F
7C87 3137 0C9F'DA
7C8A 3138
7C8A 3139 0A 3E
7C8C 3140 3D 4$:
7C8D 3141 0C8C'C2
7C90 3142
7C90 3143 0010'3A
7C93 3144 3D
7C94 3145 0010'32
7C97 3146 0C84'C2
7C9A 3147
7C9A 3148 0009'32 3C AF
7C9F 3149
7C9F 3150
7C9F 3151
7C9F 3152 20 D3 5$:
7CA1 3153
7CA1 3154 A1 D3

```

I 5

```

Fiche 1 Frame 15 Sequence 60
CALL OR_JMP_ADD ;SET UP A JUMP INSTRUCTION TO JUMP
; TO THE TOP LOCATION IN WCS.

CALL CHECK_RD ;CALL BOOT CODE IF RD IS IN CONTROL

LXI H,JMP_INST ;LOAD THE JUMP INSTRUCTION INTO
LXI D,WCS-VALUE ; THE LAST WCS LOCATION WITH
CALL MOVER_3 ; CORRECT PARITY.
LHLD WCS_SIZE
SHLD WCS_ADDRESS
SET PARITY_ON
CALL WRITE_WCS

LXI H,TEST2D_JUMP ;PREPARE TO RETURN FROM A PARITY
LXI D,PARITY_VECTOR ; ERROR.
CALL MOVER_3

CALL CHECK_RD ;CALL BOOT CODE IF RD IS IN CONTROL

LXI H,X_CLR_WRO+1 ;CLEAR WORKING REGISTER ZERO.
CALL EXEC_INST

CALL NOP_CSR ;LOAD A NOP INTO THE CSR.

LXI H,0 ;LOAD A ZERO INTO THE UPC.
SHLD WCS_ADDRESS
CALL LD_UPC

OUT ENBPARG ;ENABLE A PARITY ERROR.

OUT SETCLK ;START THE CPU CLOCK.

CLEAR NO_CPU_ATTNG ;CLEAR THIS FLAG.

MVI A,WAIT_ATTNGCNT ;LOAD THE LOOP COUNT INTO WAIT_CNT.
STA WAIT_CNT

CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL

IN CPATTNG ;UPON RECEIVING CPU ATTN...
RAR
JC 5$ ;...JUMP OUT OF WAIT LOOP.

MVI A,10 ;INCREASE WAITING TIME.
DCR A
JNZ 4$

LDA WAIT_CNT ;DECREMENT THE WAIT COUNT...
DCR A
STA WAIT_CNT
JNZ 3$ ;... AND CONTINUE LOOPING.

SET NO_CPU_ATTNG ;CPU ATTN HAS NOT BEEN RECEIVED
; IN PROPER AMOUNT OF TIME.

OUT CLRCLK ;STOP THE CPU CLOCK.

OUT DISPAR ;DISABLE PARITY ERROR.

```



```

ZZ-ENKBE-2.1  7000  4
7CA3 3155
7CA3 3156 008F'CD
7CA6 3157
7CA6 3158 0000'CD
7CA9 3159
7CA9 3160 008F'CD
7CAC 3161
7CAC 3162 00'01'21
7CAF 3163 02F8'CD
7CB2 3164
7CB2 3165 00'00'21
7CB5 3166 0300'CD
7CB8 3167
7CB8 3168 02B6'CD
7CBB 3169
7CBB 3170
7CBB 3171 008F'CD
7CBE 3172
7CBE 3173 0000'32 3C AF
7CC3 3174
7CC3 3175 0F 0009'3A
          OCD5'D2
7CCA 3176
7CCA 3177 01 3E
7CCC 3178 0000'32
7CCF 3179
7CCF 3180 044F'CD
7CD2 3181
7CD2 3182 0CEB'C3
7CD5 3183
7CD5 3184 02 3E
7CD7 3185 0000'32
7CDA 3186
7CDA 3187 00'8A'21
          02 0E 00'8D'11
          0CEB'CA 0000'CD
7CE8 3188
7CE8 3189
7CE8 3190 044F'CD
7CEB 3191
7CEB 3192
7CEB 3193
7CEB 3194
7CEB 3195
7CEB 3196 0D23'C3
7CEE 3197
7CEE 3198
7CEE 3199 20 D3
7CF0 3200
7CF0 3201 A1 D3
7CF2 3202
7CF2 3203 FB
7CF3 3204
7CF3 3205 C1
7CF4 3206
7CF4 3207 03 3E
7CF6 3208 0000'32
7CF9 3209
7CF9 3210 02 0E 00'00'21
          0D 23 77 AF

```

```

*****
*
*
*
*
*
*****
*
*
*
*
*
*****
*
*
*
*
*
*****
*
*
*
*
*
*****

```

6\$:

J 5

Fiche 1 Frame J5

Sequence 61

```

CALL CHECK_RD      ;CALL BOOT CODE IF RD IS IN CONTROL
CALL CLEAR_CPU_ATT ;CLEAR CPU ATTN.
CALL CHECK_RD      ;CALL BOOT CODE IF UNDER RD CONTROL
LXI H,X CLR_PAGE+1 ;CLEAR THE PAGE BIT.
CALL EXEC_INST
LXI H,X MOV_WRRW   ;READ WRO.
CALL LD_CSR_INST
CALL READ_DATA_16   ;READ 2 BYTES OF DATA AND STORE IN
                   ; REC_DAT.
CALL CHECK_RD      ;CALL BOOT CODE IF RD IS IN CONTROL
SET NA_OTHER        ;NO OTHER DATA.
IF NO_CPU_ATT       ;IF TIMEOUT HAS OCCURRED...

MVI A,1             ;...ERROR NUMBER 1.
STA CON_ERR_NUM
CALL TEST2D_ERROR   ;PRINT ERROR NUMBER 1.
ELSE                ;...OTHERWISE.
MVI A,2             ;CHECK FOR ERROR NUMBER 2.
STA CON_ERR_NUM
IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
                           ; RECEIVED DATA.
CALL TEST2D_ERROR   ;PRINT ERROR NUMBER 2.
ENDIF
ENDIF
JMP 7$             ;SKIP NEXT SET OF INSTRUCTIONS USED
                   ; TO RETURN FROM A PARITY ERROR.
OUT CLRCLK         ;STOP THE CLOCK
OUT DISPAR         ;DISABLE FURTHER PARITY ERRORS.
EI                 ;ENABLE FURTHER INTERRUPTS.
POP B              ;RESTORE THE STACK.
MVI A,3            ;ERROR NUMBER 3.
STA CON_ERR_NUM
ZERO CON_ADD_DAT,2 ;STORE THE CONTENTS OF THE UPC

```

```

ZZ-ENKBE-2.1 7000 4
7D05 3211 0CFF'C2
7D08 3212 0000'CD
7D0B 3213 0000'2A
7D0E 3214 0002'22
7D0E 3215 0000'CD
7D11 3216
7D11 3217 00'01'21
7D14 3218 02F8'CD
7D17 3219
7D17 3220 00'00'21
7D1A 3221 0300'CD
7D1D 3222 02B6'CD
7D20 3223
7D20 3224
7D20 3225 044F'CD
7D23 3226
7D23 3227 0000'3A
7D26 3228 0F
7D27 3229 0BDF'DA
7D2A 3230
7D2A 3231 00'00'21
7D2D 3232 4C 22 11
7D30 3233 0000'CD
7D33 3234
7D33 3235 3C D3
7D35 3236
7D35 3237
7D35 3238
7D35 3239
7D35 3240
7D35 3241
7D35 3242
7D35 3243
7D35 3244 0000'CD 51 3E
              0F 0000'3A
              C9 0D42'D2
              0F 0000'3A
              3D D3 0D4D'DA
7D4B 3245
7D4B 3246 3C D3
7D4D 3247
7D4D 3248

```

K 5

Fiche 1 Frame K5

Sequence 62

```

CALL LOAD_UPC ; CON_ADD_DAT.
LHLD UPC_VALUE
SHLD CON_ADD_DAT+2

CALL CLEAR_CPU_ATT ;CLEAR CPU ATTN.

LXI H,X CLR_PAGE+1 ;CLEAR THE PAGE BIT.
CALL EXEC_INST

LXI H,X MOV_WRR ;READ WRO.
CALL LD_CSR_INST
CALL READ_DATA_16 ;READ 2 BYTES OF DATA AND STORE IN
                  ; REC_DAT.

CALL TEST2D_ERROR ;PRINT AN ERROR MESSAGE.

7$: LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
   RRC
   JC 1$ ;...JUMP TO BEGINNING OF ERROR LOOP.

LXI H,PARITY_JUMP ;RESTORE THE PARITY ERROR RETURN
LXI D,PARITY_VECTOR ; ADDRESS.
CALL MOVER_3

END_TEST2D: OUT CLRLIT ;TURN OFF RUN LIGHT

:*****
: LAST TEST FOR EVERY CONSOL_SECTION....FORCES END OF SECTION
:*****

END_TEST: HEAD <^X51>

TAIL

```


\$PSW	= 00000006		
\$SP	= 00000006		
A	= 00000007		
ALARM_REG1	= 00000007		
ALARM_REG2	= 0000000F		
ALLOWP	= 00000003		
APT	*****	X	02
APTFLG	= 00000004		
ARM_ALL	= 0000003F		
ARM_CNTR	0000010F	R	02
ARM_CNTR_DAT	= 00000020		
AUTO	= 00000005		
B	= 00000000		
BOOTEN	= 00000007		
BOOTF	= 00000001		
C	= 00000001		
CG_PNT	0000000C	R	02
CHECK_2B_2C_ERROR	0000041D	R	02
CHECK_RD	0000008F	R	02
CHECK_REMOTE	000001EF	R	02
CLEAR_CPU_ATT	*****	X	02
CLRACK	= 000000A9		
CLRACL	= 00000033		
CLRATN	= 000000AB		
CLRBSY	= 0000002E		
CLRCLK	= 00000020		
CLRCSE	= 00000024		
CLRCSE	= 00000038		
CLRDCL	= 00000031		
CLRHLT	= 000000AF		
CLRLIT	= 0000003C		
CLRMCI	= 00000029		
CLRMCK	= 0000002B		
CLRPFI	= 000000AD		
CLRSS	= 00000022		
CLRTIM	= 000000A5		
CLRTMR	= 0000002C		
CLRUPC	= 000000A8		
CLRUPK	= 00000026		
CLRWC	= 00000036		
CLR.CG_PNT	000000B3	R	02
CLR.CPU_ATT	00000049	R	02
CLR_OUTPUT	0000016C	R	02
CLR_OUTPUT_DAT	= 000000E0		
CLR_SER_OUT	= 00000040		
CNT	= 00000802		
CNTA	= 00000FA0		
CNTLC_TYPED	*****	X	02
COMPARE	*****	X	02
CON_ADD_DAT	*****	X	02
CON_BEGIN_TEST	*****	X	02
CON_ERROR	*****	X	02
CON_ERR_NUM	*****	X	02
CON_EXP_DAT	*****	X	02
CON_MOD_DAT	*****	X	02
CON_MSK_DAT	*****	X	02
CON_REC_DAT	*****	X	02

COUNT_SAVE	00000011	R	02
CPATTN	= 00000083		
CPUACK	= 00000082		
CR	*****	X	02
CSR07	= 00000085		
CSR15	= 00000086		
CSR23	= 00000087		
CSR_SHIFT	*****	X	02
CSR_VALUE	*****	X	02
D	= 00000002		
DCLO	= 00000002		
DECR_WORD	*****	X	02
DISARM_CNTR	00000124	R	02
DISARM_CNTR_DAT	= 000000C0		
DISMEM	= 000000A7		
DISPAR	= 000000A1		
DOLOOP	*****	X	02
E	= 00000003		
ENBMEM	= 000000A6		
ENBPAR	= 000000A0		
END_TEST	00000D35	R	02
END_TEST2D	00000D33	R	02
ENTRY	00000000	R	02
EOS_FLG	*****	X	02
ERROR_CON	*****	X	02
EXEC_INST	000002F8	R	02
EXP_DAT	0000008A	R	02
EXP_SHIFT	00000089	R	02
FILE_NAME	00000005	R	02
GCVECT	*****	X	02
H	= 00000004		
HEAD	= 00000000		
HEADER_B	00000022	R	02
HEX_BUF	*****	X	02
HLTFLG	= 00000002		
HOLD_REG	= 00000010		
INC.CG_PNT	000000B8	R	02
INC_CNTR	00000175	R	02
INC_CNTR_DAT	= 000000F0		
INC_WORD	*****	X	02
INC_WRO	00000043	R	02
INITH	= 00000035		
INITL	= 00000034		
ITCSR	= 0000001D		
ITDB	= 0000001C		
JMP_INST	0000004C	R	02
L	= 00000005		
LAT100HZ	= 00000006		
LD_CSR_INST	00000300	R	02
LD_UPC	*****	X	02
LITERAL	= 00000001		
LOAD_CNTR	00000139	R	02
LOAD_CNTR_DAT	= 00000040		
LOAD_CSR	*****	X	02
LOAD_REG	= 00000008		
LOAD_UPC	*****	X	02
LVL	= 00000000		

ZZ-ENKBE-2.1 Symbol table
MAIN.
Symbol table

M 5
24-FEB-1983

Fiche 1 Frame M5

Sequence 64

24-FEB-1983 15:27:14 VAX-11 Macro V03-00
24-FEB-1983 15:25:46 DRBO:[GREEN.WCS]ENKBE.MAC;57 Page 100
(1)

M	= 00000006		
MACSTK1	= 00000802		
MACSTK2	= 000007FF		
MACSTK3	= 000007F2		
MACSTK4	= 000007EC		
MASTER_MODE	= 00000017		
MASTER_RESET	= 000000AE	R	02
MIPFLG	= 000040D9		
MISC2	= 00000001		
MODE_REG	= 00000000		
MOVER	*****	X	02
MOVER_3	*****	X	02
MWCS_A	*****	X	02
NA_EXP_REC	*****	X	02
NA_OTHER	*****	X	02
NOP_CSR	*****	X	02
NO_CPU_ATTN	= 00000009	R	02
OKV15	= 00000007		
ONE_DAT	= 00000053	R	02
ONE_SHIFT	= 00000050	R	02
OR_JMP_ADD	= 00000333	R	02
PARITY_JUMP	*****	X	02
PARITY_ON	*****	X	02
PARITY_VECTOR	= 00004C22		
POWER_VECTOR	= 00004C25		
PRINT_HEX	*****	X	02
PRINT_STRING	*****	X	02
READ	= 00000080		
READJ1	= 00000003		
READJ2	= 00000004		
READ_DATA_16	= 000002B6	R	02
READ_HOLD	= 000000DD	R	02
READ_ITDB	= 000000FB	R	02
REC_DAT	= 0000008D	R	02
REC_DONE	= 00000002		
REC_SHIFT	= 0000008C	R	02
REMDIS	= 00000006		
REM_REC_BIT	= 00000020		
REM_TX_BIT	= 00000002		
REPORT_ERROR	= 0000037B	R	02
REPORT_ERROR_2	= 00000369	R	02
RESET_DATA	= 000000FF		
RESTORE_TR	= 000001E7	R	02
RESTORE_TT	= 000001D7	R	02
RESTORE_TU	= 000001DF	R	02
SAVE_CNTR	= 0000014E	R	02
SAVE_CNTR_DAT	= 000000A0		
SAVE_WCS_ADD	= 00000007	R	02
SETACK	= 000000A8		
SETACL	= 00000032		
SETATN	= 000000AA		
SETBSY	= 0000002F		
SETCLK	= 00000021		
SETCSK	= 00000025		
SETCSR	= 00000039		
SETDCL	= 00000030		
SETHLT	= 000000AE		

SETLIT	
SETMCI	
SETMCK	
SETPFI	
SETSS	
SETTIM	
SETTMR	
SETUPC	
SETUPK	
SETWCK	
SET_CPU_ATTN	
SET_FOR_CONT	
SET_OUTPUT	
SET_OUTPUT_DAT	
SET_SER_OUT	
SET_TR_CB	
SET_TT_LB	
SET_TU_LB	
SHIFTER	
SHIFT_CNT	
SHIFT_PNT	
SKIP_TEST	
SKIP-UA1	
SKIP-UA3	
SUMREG	
SYM	
TEMP_DAT	
TEMP_SHIFT	
TEMP_WORD	
TER_REC_BIT	
TER_TX_BIT	
TEST27	
TEST27_DAT	
TEST27_ERROR	
TEST28	
TEST28_DAT	
TEST28_ERROR	
TEST29	
TEST29_DAT	
TEST29_ERROR	
TEST2A	
TEST2A_2_ERROR	
TEST2A_DAT	
TEST2A_ERROR	
TEST2B	
TEST2B_2C_ERROR	
TEST2B_DAT	
TEST2C	
TEST2D	
TEST2D_ERROR	
TEST2D_JUMP	
TEST2D_PAR_ERR	
TIMER_DAT	
TRCR	
TRDB	
TRMODE	
TRSR	

= 0000003D		
= 00000028		
= 0000002A		
= 000000AC		
= 00000023		
= 000000A4		
= 0000002D		
= 000000A9		
= 00000027		
= 00000037		
00000046	R	02
*****	X	02
00000163	R	02
= 000000E8		
= 000000C0		
000001C9	R	02
0000019E	R	02
000001BB	R	02
*****	X	02
0000000D	R	02
*****	X	02
*****	X	02
0000000A	R	02
0000000B	R	02
= 00000020		
= 00000802		
00000087	R	02
00000086	R	02
00000012	R	02
= 00000008		
= 00000001		
0000045E	R	02
00000057	R	02
0000038A	R	02
00000551	R	02
0000005D	R	02
000003AD	R	02
000005D2	R	02
00000061	R	02
000003D2	R	02
0000070B	R	02
00000407	R	02
0000006D	R	02
000003F3	R	02
00000836	R	02
0000042F	R	02
0000007B	R	02
00000975	R	02
00000BA5	R	02
0000044F	R	02
00000083	R	02
00000BC1	R	02
0000000E	R	02
= 00000043		
= 00000040		
= 00000042		
= 00000041		

ZZ-ENKBE-2.1 Symbol table
MAIN.
Symbol table

N 5
24-FEB-1983

Fiche 1 Frame N5

Sequence 65

24-FEB-1983 15:27:14

VAX-11 Macro V03-00

Page 101

24-FEB-1983 15:25:46

DRB0:[GREEN.WCS]ENKBE.MAC;57

(1)

TTCR	=	0000004B		
TTDB	=	00000048		
TTMODE	=	0000004A		
TTSR	=	00000049		
TU58_REC_BIT	=	00000010		
TU58_TX_BIT	=	00000004		
TUCR	=	00000047		
TUDB	=	00000044		
TUMODE	=	00000046		
TUSR	=	00000045		
TX_RDY	=	00000001		
UATT_RESTORE	=	00000035		
UA_LOOPBACK	=	000000B3		
UA_RESTORE	=	00000015		
UPC14	=	00000084		
UPC14A	=	00000000		
UPC_VALUE		*****	X	02
VERSION		00000003	R	02
VNORML	=	000000A3		
VSHIFT	=	000000A2		
WAIT_ATTN_CNT	=	000000FF		
WAIT_CNT		00000010	R	02
WAIT_CNT1	=	00000045		
WAIT_CNT2	=	00000050		
WAIT_CNT3	=	000000C0		
WAIT_LOOP		0000017E	R	02
WCSB0	=	00000008		
WCSB1	=	00000009		
WCSB2	=	0000000A		
WCS_ADDRESS		*****	X	02
WCS_ADDR_PNT		00000014	R	02
WCS_READ		*****	X	02
WCS_SIZE		00000016	R	02
WCS_SIZER		00000212	R	02
WCS_SIZER_ADDR		00000018	R	02
WCS_SIZE_P		0000029F	R	02
WCS_VALUE		*****	X	02
WCS_WRITE		*****	X	02
WRITE	=	000000CC		
WRITE_ALARM		000000E9	R	02
WRITE_ITDB		00000106	R	02
WRITE_LOAD		000000D1	R	02
WRITE_MASTER		000000BD	R	02
WRITE_MODE		000000C5	R	02
WRITE_WCS		*****	X	02
X_CLR_PAGE		*****	X	02
X_CLR_WRO		*****	X	02
X_MOV_WRRR		*****	X	02
X_SET_PAGE		*****	X	02
X_SHIFT_R		*****	X	02
Y	=	00000050		
YBUSRD	=	000000EC		
ZERO_DAT		0000004F	R	02
ZERO_SHIFT		00000054	R	02

ZZ-ENKBE-2.1 Psect synopsis
.MAIN.
Psect synopsis

B 6
24-FEB-1983

Fiche 1 Frame B6 Sequence 66
24-FEB-1983 15:27:14 VAX-11 Macro V03-00
24-FEB-1983 15:25:46 DRB0:[GREEN.WCS]ENKBE.MAC;57 Page 102
(1)

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
CPU5	00000D4D (3405.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	21	00:00:00.05	00:00:00.46
Command processing	32	00:00:00.25	00:00:00.95
Pass 1	959	00:00:29.21	00:01:43.48
Symbol table sort	0	00:00:00.42	00:00:01.40
Pass 2	535	00:00:11.96	00:00:30.68
Symbol table output	2	00:00:00.17	00:00:00.44
Psect synopsis output	3	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1556	00:00:42.10	00:02:17.47

The working set limit was 1000 pages.
280295 bytes (548 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 282 non-local and 116 local symbols.
5272 source lines were read in Pass 1, producing 41 object records in Pass 2.
151 pages of virtual memory were used to define 150 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
SYS\$SYSROOT:[SYSLIB]STARLET.MLB;1	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKBE.LIS/OBJECT=ENKBE.OBJ 8085MAC.MAR+MACDEF.MAC+[JENKBE.MAC